commodore Magazine

AÑO I - Núm. 3 - Mayo 1984 - 250 Ptas.

REVISTA INDEPENDIENTE PARA USUARIOS



Magic Desk, el despacho en casa

Herramientas para el programador

Interfaces para todos

commodore *Magazine*

Sumario

Commodore Magazine es una publicación de Ediciones y Suscripciones S.A., C/Bravo Murillo, 377 - Madrid 20, Tel. (91) 733 74 13 / 47 / 63 / 97.

REDACCION
Director:
Alejandro Diges.
Colaboradores:
Anibal Pardo.
Gumersindo García.
Roberto Menéndez.
Simeón Cruz.
Miguel Angel de Frutos.
Manuel Arias.
Diseño:
Ricardo Segura.

EDITORIAL
Presidente:
Fernando Bolín.
Director Editorial:
Norberto Gallego.

ADMINISTRACION
Gerente de Circulación y Ventas: Luis Carrero.
Suscripciones: Antonio Zurdo.
Producción:
Miguel Onieva.
Publicidad Madrid:
Roberto Rodriguez.
Bravo Murillo, 377.
Madrid - 20.
Tel. (91) 733 74 13.
Publicidad Barcelona:
Enrique Alier.
Pelayo, 12.
Tel. (93) 301 47 00, Ext. 27
Distribuye: SGEL. Avda. Valdelaparra s/n, Alcobendas, Madrid.
Imprime: Novograph S.A., Ctra. de Irún, Km 12.450 Madrid.
Fotomecánica: Karmat. Pantoja, 10, Madrid.
Depósito Legal: M-6622-1984

Año 1
Num. 3

Editorial

Parece que fue ayer cuando con toda ilusión salimos a la calle. Sin embargo ya estamos con el número 3. Las cartas de felicitaciones (también alguna queja) inundan las mesas de la redacción. Los participantes en el concurso de software *forman un verdadero aluvión. Por el contrario, la participación en el concurso de aplicaciones para el* Calc-Result *está siendo tímida en exceso. Hay que animarse, el premio merece la pena y ni tan siquiera es preciso disponer del programa para pensar la aplicación.*

La cantidad de artículos e ideas que tenemos pendientes de publicar es amplia. Este mes, por ejemplo, tampoco hemos encontrado lugar para el montaje electrónico. Será el número que viene, seguro que gustará a muchos poseedores del Vic-20.

Aunque el interés y calidad de muchos artículos pendientes están asegurados, no nos hemos atrevido a robarle espacio a las secciones dedicadas a Programas y Concurso. Suponemos que son dos de las secciones con mayor aceptación por vuestra parte. No obstante nos gustaría conocer vuestra opinión. Iniciamos en esta edición una serie de artículos destinados a los principiantes que se encuentran perdidos en la lectura del manual. La intención es hacerla muchos más amena y comprensible.

Adelantabamos hace un mes que estaríamos en la Feria de Hannover, para conocer las últimas novedades de Commodore. *Efectivamente, así ha sido y os lo contamos en las páginas correspondientes. Tener por seguro que estamos deseando disponer en nuestras manos algunos de estos productos, para contaros si son o no lo fantásticos que parecen.*

Una gran mayoría de las llamadas que haceis, o de cartas que nos escribís, hacen referencia a supuestos fallos en la reproducción de los programas en la revista. Tener por seguro que todos, absolutamente todos, los programas han funcionado antes de ser publicados. En algún caso, como en el Dragón Temible, *por un error en el montaje habían quedado fuera varias líneas, (que añadimos en este número). La mayoría de los supuestos fallos se eliminan repasando con mucho cuidado lo que hemos tecleado. Hacerlo así, pero si persiste algún problema, no dudeis en contactarnos.*

Claves para introducir en el ordenador los programas que aparecen en la revista.

COMO SE VE	COMO SE TECLEA	EFECTO CONSEGUIDO
	SHIFT + CLR	(LIMPIA PANTALLA)
	CLR	(HOME)
	CRSR	(CURSOR ABAJO)
	SHIFT + CRSR	(CURSOR ARRIBA)
	CRSR	(CURSOR DERECHA)
	SHIFT + CRSR	(CURSOR IZQUIERDA)
	CTRL + 1	(NEGRO)
	CTRL + 2	(BLANCO)
	CTRL + 3	(ROJO)
	CTRL + 4	(CIAN)
	CTRL + 5	(VIOLETA)
	CTRL + 6	(VERDE)
	CTRL + 7	(AZUL)
	CTRL + 8	(AMARILLO)
	CTRL + 9	(CARACTER INVERSO)
	CTRL + 0	(CARACTER NORMAL)

Ma
Des


MAGIC DESK I
(C) 1983
BY COMMODORE


MAGIC DESK I
(C) 1983
BY COMMODORE
0:02

agic
k

el despacho en casa

En los últimos tiempos se está poniendo de moda hablar sobre la automatización de la oficina. En un futuro no muy lejano ya no será necesario desplazarse a un lugar común de trabajo. La posibilidad de trabajar en casa, efectuando todo tipo de comunicaciones con las personas y los datos que necesitemos será algo sumamente sencillo. Además la productividad alcanzará cotas difícilmente imaginables, y el esfuerzo empleado para llevar a cabo las mismas tareas, será increíblemente menor. El terminal inteligente, altamente interactivo, va a ser todo el requisito necesario. Por supuesto, la cantidad de papel empleado disminuirá en favor de las pantallas de muy alta resolución y las unidades de almacenamiento masivo, con alto índice de empaquetamiento y costo ridículo.

Aunque mucha de la tecnología necesaria ya está presente en la industria informática, todavía tendrá que bajar mucho el precio y la mentalidad deberá cambiar en una medida paralela, para que el usuario medio pueda acceder a ella.

Sin embargo, el feliz propietario del **Commodore 64** ya puede beneficiarse de estos adelantos. Hace muy poco tiempo, **CBM** ha presentado una familia de cartuchos, bautizados genéricamente como **Magic Desk** (El pupitre mágico), que posibilita la materialización del despacho electrónico. Las imágenes que aparecen en pantalla son de lo más elocuentes. No cabe la menor duda de que todo es tan familiar como pueda serlo nuestra mesa de trabajo.

El manejo de este *software* es totalmente intuitivo. Para trabajar con él es preciso disponer de un *joystick*, que es el encargado de mover un dedo indicador por la pantalla, a nuestro capricho.

Una vez conectado el cartucho, nada más encender el ordenador, aparece en pantalla un mensaje de saludo, presidido por el anagrama de **Commodore.** Inmediatamente aparece la representación visual de un despacho, tal como ilustra la foto. En él se pueden ver varias cosas, tales como una máquina de escribir, un libro de contabilidad, un teléfono, una calculadora, una agenda, un fichero de tres cajones, con el reloj encima, la papelera y, por supuesto, la mesa.

Conviene aclarar que el **Magic Desk** completo está incluido en varios cartuchos, pero por el momento sólo hemos podido disponer del I, que soporta las funciones de escritura de cartas y documentos y manejo del fichero.

El dedo puede moverse libremente por toda la pantalla, de manera solidaria con el movimiento del *joystick*. Lo más aconsejable en primer lugar es poner en hora el reloj. Señalándolo, se aprieta el pulsador de fuego del *joystick*, apareciendo un recuadro blanco en torno al mismo. Esto indica que podemos introducir la hora en forma de 4 dígitos decimales. Una vez conformes, se aprieta por segunda vez el pulsador, desaparece el recuadro y el dedo queda liberado para continuar el movimiento. A continuación podríamos querer escribir una carta. Nada más sencillo. Nuevamente con el movimiento de la mano indicamos nuestro deseo. Tras presionar el pulsador, tenemos ante nosotros lo que muy bien podría ser el carro de una máquina de escribir clásica, en la parte superior. Compartiendo la pantalla se hallan las representaciones iconográficas de los elementos que se pueden necesitar: el dedo, la impresora, la papelera, la tabulación, el despacho —para volver a la vista general— y la propia máquina, que aparece con el típico recuadro blanco. Para acceder a cualquiera de estas posibilidades, basta con apretar el pulsador que hace desaparecer el recuadro en torno a la máquina y liberar el movimiento de la mano. Nuevamente el procedimiento de señalar nos llevará a lo que deseamos hacer.

Supongamos que necesitamos establecer marcas tabuladoras. En principio solamente podemos desplazar el carro entre las marcas de 10 y 70 caracteres, pero podemos ampliar el margen entre 0 y 80. Para efectuar estos cambios, señalaremos la imagen correspondiente, situada al pie de la máquina, presionando el consabido botón. Se establece el margen izquierdo mediante el movimiento del carro con el *joystick*, presionado nuevamente el botón. Igualmente se hace con el derecho. Para saltarnos los márgenes en cualquier circunstancia durante la escritura, basta con presionar la tecla F1/F2 del ordenador.

Con la tecla F3/F4 se pueden establecedr varias marcas tabulado-

ras. Disponiendo el carro en el lugar donde queremos que permanezca la marca, se presiona F5/F6. Utilizando posteriormente F3/F4 el carro se desplaza hasta esa posición. Borrar las marcas es una tarea que se acomete con F7/F8.

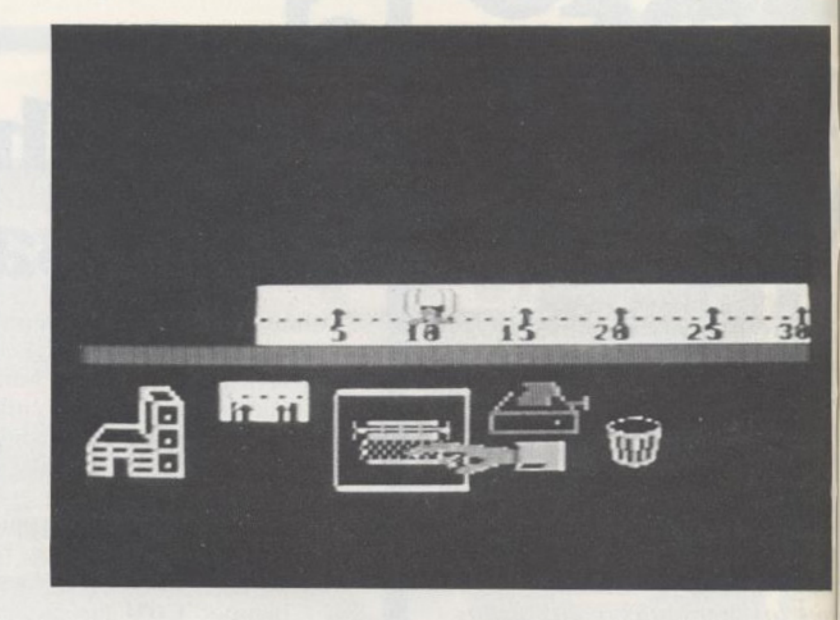
En las ilustraciones adjuntas decidimos escribir una carta a un conocido. El procedimiento es totalmente similar a escribir con una máquina clásica. Return devuelve el carro automáticamente al principio de línea y pasa a la línea siguiente. La barra espaciadora funciona exactamente igual que su homóloga mecánica. Las mayúsculas y minúsculas no representan dificultad adicional. Quizás una de las posibilidades más interesantes es que nos podemos equivocar cuantas veces queramos. Las teclas de vuelta hacia atrás y avance, en realidad la posibilidad de mover el cursor en pantalla, permite reescribir donde estuviera el error.

Una opción importante que permite este cartucho es pedir ayuda cuando sea necesaria. Simplemente con presionar la tecla que lleva inscrito el anagrama de **Commodore,** aparece el menú correspondiente a la fase del programa en que nos encontramos. Los menús son totalmente intuitivos, puesto que se sigue conservando la idea de la mano que señala y las imágenes alegóricas de cada posibilidad.

Una vez terminada la carta, señalamos la impresora. Simplemente con presionar el botón del *joystick*, el trabajo de impresión comienza sobre el papel. Aquí podremos utilizar la impresora matricial o una de margarita para obtener mayor calidad en la escritura. Una nueva copia de la carta, tal vez cambiando la fecha o el nombre de la persona a quien va dirigida, si fuera una circular, solamente requiere efectuar los mínimos cambios en la pantalla y presionar nuevamente el botón. El trabajo que puede ahorrarse es notable, con respecto al método tradicional.

Cuando un documento escrito deja de tener valor, lo más normal es que lo tiremos a la papelera. Pues bien la papelera que vemos en la pantalla no es simplemente un adorno, es una papelera real. Solamente con señalar su imagen, en cualquiera de la pantalla donde aparece, basta con presionar el botón. Pero podemos arrepentirnos. Con la primera pulsación el documento se sitúa al lado de la papelera, para asegurar que deseamos borrarlo de la memoria del ordenador, hay que presionar una segunda vez. Un simpático sonido indica que el papel ha pasado a mejor vida.

Por el contrario, podríamos desear que el documento quedase archivado. Señalemos pues el fichero y en pantalla aparecen las carpetas correspondientes al mismo. Como es lógico, las labores de fichero son acometidas por la unidad de *diskettes*. Un *diskette* se compone en este caso de tres cajones, cada uno con diez carpetas, en las que





pantalla una representación de las páginas. La forma de buscar es igual a la empleada para seleccionar carpetas. Una vez seleccionada, su contenido es cargado desde el *diskette* a la memoria central del ordenador y podremos trabajar con ella dentro de las posibilidades que ofrece el cartucho.

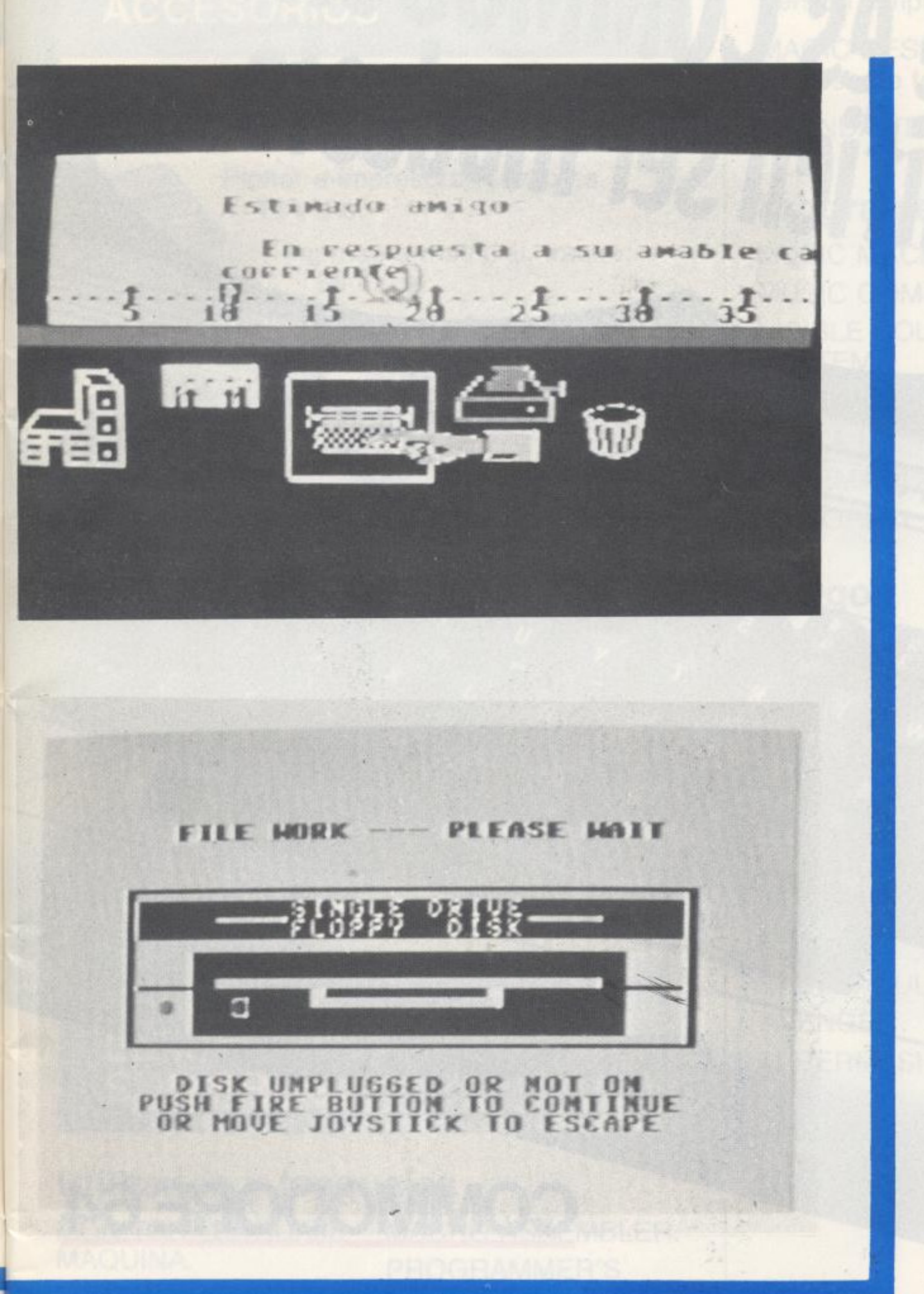
Como es obvio, los *diskettes* utilizados deben estar formateados desde un principio. Esto es importante, pues supongamos que después de haber tecleado un complejo documento nos damos cuenta de que no tenemos *diskettes* inicializados.

En cualquier circunstancia esto nos obligaría a desconectar el ordenador, después el cartucho, formatear el *diskette* y volver a teclear el documento. Pero esto no es así, el **Magic Desk** permite efectuar la tarea sólo señalando con el dedo. Parece una tontería, pero podría traernos algún enfado si no se hubiera previsto la posibilidad.

Cuando algo no está como debiera ser, por ejemplo que quisiéramos imprimir y no estuviera conectada la impresora o necesitáramos archivar y no estuviera presente la unidad de *diskettes*, una señal de prohibido aparece exactamente sobre una representación gráfica del periférico en cuestión. La primera idea que surge cuando todavía no se ha hincado el diente al **Magic Desk** es que estamos ante un juguete sofisticado. Sin embargo, la opinión se desvanece después de un rato de trabajar con él. Su sencillez de manejo y la excelente calidad gráfica, le convierten en útil herramienta de trabajo para quienes gusten de mecanizar sus procedimientos de trabajo. Su potencia es suficiente para muchas funciones domésticas o de generación de cartas y circulares, así como documentos, pero sería poco realista pretender automatizar una oficina con gran volumen de trabajo.

Esperamos en un futuro no muy lejano poder comentar los restantes módulos que componen el **Magic Desk**. Será en cuanto los tengamos

La redacción



a su vez se pueden almacenar hasta diez páginas. El total archivable en un *diskette* corresponde a unas treintas páginas de texto. Las carpetas, de color amarillo, y las páginas, de color blanco, pueden ser etiquetadas individualmente y cambiadas cuando sea necesario. Para buscar, la selección de carpetas se hace moviendo en la pantalla hacia adelante o atrás, como en un fichero corriente, desplazando el *joystick* arriba y abajo, hasta seleccionar una.

La carpeta se abre después de desactivar la pantalla de las carpetas, con una presión del botón del *joystick*. Después se hace que el dedo señale una imagen en la que aparecen tres hojas superpuestas y se vuelve a presionar el botón. Es en este momento cuando la unidad de *diskettes* denota la búsqueda y aparece en la

COMMODORE 64 LE DA ACCESO A MUCHOS ACCESORIOS

Unidad simple de disco (Monofloppy) 170 K.

Cassette.



Plotter e impresora, 4 colores, 14 c.p.s.

Impresora matricial, tractor, 30 c.p.s.

Interface RS232.

Joy Stick.

Paddle.

Cursos de Introducción al BASIC.

COMMODORE 64 LE MUESTRA PARTE DE SUS PROGRAMAS

Utilitarios y lenguajes

MONITOR LENGUAJE MAQUINA.	MACRO ASSEMBLER.
FORTH.	PROGRAMMER'S UTILITIES.
LOGO.	TURTLE GRAPHICS II.
PILOT.	MASTER.

Sistemas operativos

FILE/BOSS.	CP/M.

Programas de aplicaciones

EASY SCRIPT.
Proceso de texto de gran potencia.

CALC RESULT.
Hoja electrónica de cálculo.

EASY CALC RESULT.
Versión simplificada del CALC RESULT.

MAGIC DESK.
Proceso de texto y gestión de ficheros.

AGENDA TELEFONICA.

Programas educativos

MUSIC MACHINE.	GEOGRAFIA I.
MUSIC COMPOSER.	GEOGRAFIA II.
VISIBLE SOLAR SYSTEM.	JUEGOS EDUCATIVOS.
SPEED/BINGO MATH.	TEMAS MONOGRAFICOS.
FISICA I.	CONOCIMIENTOS GENERALES.
MATEMATICAS I.	
HISTORIA I.	QUIMICA I.

Juegos

JUPITER LANDER.	FROGMASTER.
KICKMAN.	GRID RUNNER.
SEAWOLF.	ATTACK OF THE MUTANT CAMELS.
RADAR RAT RACE.	
TOOTH INVADERS.	
LAZARIAN.	THE PIT.
OMEGA RACE.	MR. TNT.
LE MANS.	6 GAME PROGRAMS.
PINBALL SPECTACULAR.	BINGO.
	ROOTING TOOTING.
AVENGER.	MINESSOTA FAT'S POOL CHALLENGE.
SUPERMASH.	

... y seguimos ampliando la lista

El ordenador personal de la familia más potente



MICROELECTRONICA Y CONTROL, S.A.
c/. Taquígrafo Serra, 7, 5.º. Barcelona-29
c./ Princesa, 47, 3.º G. Madrid-8

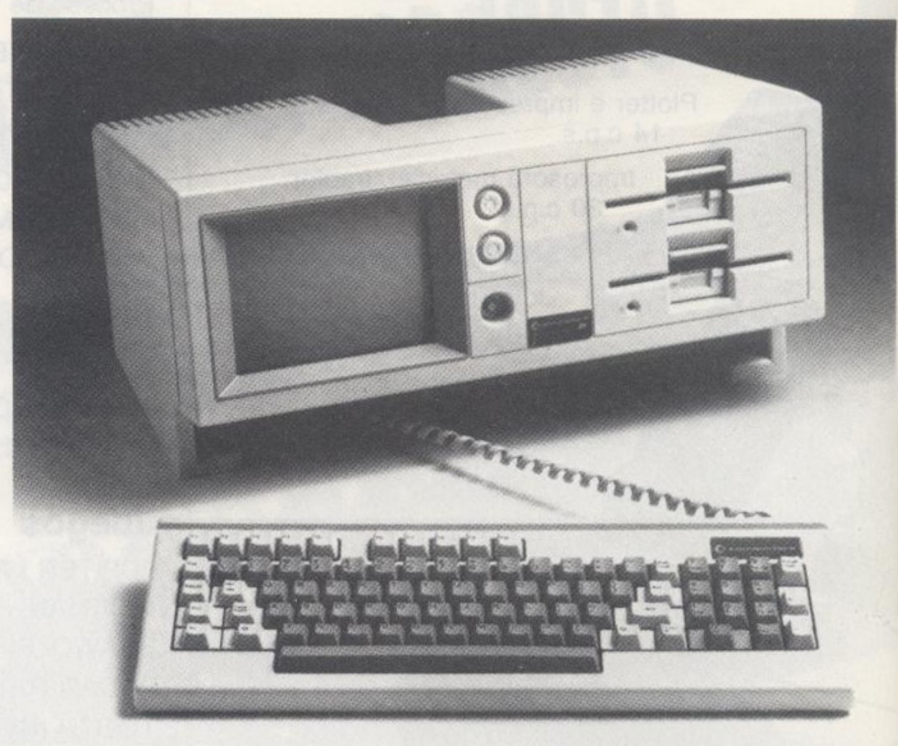
Lo prometido es deuda. Anticipábamos en el número anterior nuestra presencia en la **Feria de Hannover,** el más importante evento informático europeo. En la misma **Commodore International** ofreció una concurrida rueda de prensa. La expectación era enorme. La presencia de la plana mayor de la firma hacía presagiar el anuncio de interesantes productos, como de hecho así fue.

El flamante presidente de la compañía, **Marshall F. Smith,** hizo una exposición de los caminos que emprenderá **Commodore** bajo sus recién tomadas riendas.

Por su lado el vicepresidente, **Harold Speyer,** abrumó a todo el mundo con su exposición de cifras, tendencias y perspectivas.

En lo que a productos se refiere, hubo de todo. La gama presentada cubre un amplio abanico del mercado de usuarios. Tal vez el anuncio más sorprendente y menos esperado fue el de los modelos **C 16** y **C 116.** El primero es un ordenador doméstico de apariencia similar al **CBM 64,** con 16 Kbytes de RAM y 32 Kbytes de ROM, exactamente igual que el C 116. Ambos utilizan la versión interpretada del BASIC 3.5 y el microprocesador 7501. En realidad, la mayor diferencia que separa a los dos es la carcasa, el 116 utiliza una prácticamente igual a la que aloja al **C 264,** que también fue presentado aquí para Europa (en el n.° 1 ofrecimos la información sobre este equipo). El formato de pantalla elegido es más estándar que en modelos previos, con 25 líneas de hasta 40 caracteres. En modo gráfico la resolución es de 200 por 320 puntos. El sintetizador musical incorporado sigue siendo una constante. En cuanto a comunicaciones dispone de *interface* serie, dos *ports* para *joysticks,* así como conectores para el *cassette,* salida de video y televisor.

De cara a la pequeña y mediana empresa se presentó el **CBM 8296,** cuya carcasa está en la línea de la empleada en las familias **700** y **8000.** En realidad se trata de un avance de esta última. Su memoria RAM es de 128 Kbytes, distribuidos en dos bancos de 64 K cada uno. Cuando es conectado, resulta activado el banco



principal con el correspondiente BASIC 4.0 en ROM. La ROM dispone de capacidades totales o parciales para conmutar la RAM. El sistema operativo utiliza los 32 Kbytes de RAM más bajos como memoria principal. En lugar de utilizar los 32 K superiores, también se puede recurrir a la mitad del segundo banco. Esta posibilidad es reconocida por el sistema operativo **LOS-96,** que puede ser cargado desde el diskette en los 32 K inferiores.

Los lenguajes de programación que puede emplear son el **TCL Pascal, UCSD-Pascal, COMAL** y **Ensamblador.**

Para conexión con el mundo exterior, el **CBM 8296** dispone del bus IEEE-488, el *port* de usuario, con E/S serie y señales de video para conexión de una pantalla. Igualmente aparecen dos *ports* para *cassette.*

El **CBM Z 8000** es uno de los sistemas grandes más interesantes que pudimos ver. Se trata de un sistema multiusuario, que incorpora el potente microprocesador Z 8000 de **Zilog** (una versión muy avanzada en 16 bits del Z 80). En la versión estándar, el sistema puede soportar dos terminales, pero en un futuro se asegura que podrá hacerlo con ocho o más. Al parecer se trata de la primera piedra sobre la que se levantará una familia mayor.

Entre las características del **CBM Z 8000** encontramos 256 Kbytes de RAM y 32 Kbytes de ROM y 128 Kbytes de RAM destinada a la pantalla, lo que permite que la resolución gráfica de la pantalla alcance los 1024 por 1024 puntos. Dispone del *interface* IEEE/84 y de otro tipo Centronics, así como de dos serie RS-232 programables. El coprocesador aritmético Z 8070 es opcional, pero permite mayor velocidad de ejecución en este tipo de cálculos.

El sistema operativo empleado es el **COHOS,** una adaptación del popular **UNIX.** Los lenguajes de programación utilizables por el momento son **BASIC, Pilot, Ensamblador** y **compilador C.**

Para el almacenamiento masivo, se utilizan la unidad doble de *diskettes,* con capacidad para 1,3 Mbytes y DMA (acceso directo a memoria), para mayor velocidad de acceso. También existe la opción de 10 Mby-

Noticias y Novedades



tes de almacenamiento en unidad tipo Winchester.

Para terminar, el mas esperado de los anuncios fue el correspondiente al compatible con el **PC** de **IBM**, que anunciábamos en el editorial del número anterior. Se trata en realidad del microordenador canadiense de nombre **Hyperion**. Por lo que pudimos ver, no se habia cambiado tan siquiera el color de la carcasa, solamente aparecia una pequeña plaquita con el nombre y anagrama de **Commodore**. Este modelo portátil ha sido uno de los contendientes más importantes de **IBM** en los Estados Unidos. Lleva el monitor incorporado en la carcasa, así como dos unidades de *diskette*. El teclado es separado, pero puede ser alojado en la base del ordenador.

Finalmente, dos nuevas impresoras se incorporan a la oferta de **Commodore**. El modelo **DPS 1101** es una impresora de margarita, con impresión en modo bidireccional, una velocidad máxima de impresión de 18 caracteres por segundo, espaciado proporcional, dos copias más original, si es necesario, y posibilidad de conexión con los modelos **CBM 64**, **Vic 20** y **C 264**.

La otra es una impresora matricial de color, que lleva la denominación **MCS 801**. Puede imprimir hasta en siete colores, con una velocidad máxima de 50 caracteres por segundo. Puede imprimir en letra mayúscula y minúscula, además de disponer de caracteristicas gráficas.

La presentación ha sido de lo más completa.

La otra forma de leer el manual del CBM 64

Conocer la forma de manejar las cadenas de caracteres se revela útil en extremo a la hora del desarrollo de cierto tipo de programas de aplicación. Su mayor conocimiento se traduce en mayor efectividad.

JUEGO DE CARACTERES Y MANEJO DE CADENAS EN EL 64

CODIGOS DE CARACTER

El juego de caracteres del **Commodore 64** puede considerarse como su único alfabeto consistente en 256 elementos, que incluyen caracteres alfabéticos y numéricos, gráficos en forma de pixels, signos de puntuación, símbolos matemáticos, caracteres de control, etc. Cada elemento se distingue por su código de carácter, que es un único valor en el rango de cero a 255.

El **Commodore 64** incorpora dos funciones que permiten al usuario la obtención de los códigos de carácter a partir del juego de caracteres y viceversa.

ASC. Cuando se aplica **ASC** a una expresión o cadena de caracteres se obtiene el código de carácter del primer carácter de la cadena.

sintaxis: ASC (cadena)
ejemplo: 10 X= ASC ("ABCDE") asigna el valor 65 a X, ya que el código de carácter de A es 65.

CHR$. Cuando se aplica **CHR$** a un número en el rango de cero a 255 la función devuelve el carácter cuyo código de carácter es ese número.

sintaxis: CHR$ (número)
ejemplo: 10 X$ = CHR$ (65) asigna el carácter A a X$.

El programa 1 presenta en pantalla los códigos de carácter del **64** con la excepción de los caracteres de control, que como se verá afectan a la pantalla.

Otras tres funciones para el manejo de cadenas son:

LEN. Aplicada a una cadena proporciona el número de caracteres de dicha cadena.

sintaxis: LEN (cadena)
ejemplos: LEN ("COMMODORE") da como resultado 9
LEN ("") da como resultado 0

STR$. Aplicada a un número lo convierte en una cadena de caracteres que representan a dicho número en formato de impresión.

sintaxis: STR$ (número)
ejemplos: STR$ (12.01) origina la cadena "12.01"
STR$ (—12.01) origina la cadena "—12.01"
STR$ (100.000) origina la cadena "100"
STR$ (1000000000) origina la cadena "1 E + 9"

VAL. Aplicada a una cadena de caracteres devuelve el número que existe empezando por el carácter de la izquierda. Se ignoran los caracteres no numéricos después del número. Si el primer carácter es no numérico, la función devuelve el valor 0.

sintaxis: VAL (cadena)
ejemplos: VAL ("100") proporciona el valor 100
VAL ("R101") proporciona el valor 0

FUNCIONES DE MANEJO DE CADENAS DE CARACTERES

El BASIC del **Commodore 64** dispone de tres funciones para el manejo de cadenas, muy valiosas en el manejo de caracteres y texto. Estas funciones son **LEFT$, RIGHT$** y **MID$** y se utilizan para extraer una subcadena de otra cadena.

LEFT$ (A$,N) y RIGHT$ (A$,N) extraen N caracteres a partir de los extremos izquierdo y derecho de la cadena respectivamente.

Ejemplos:
Si A$ = "ABCDEFGHIJ" entonces LEFT$ (A$,3) da lugar a "ABC"
RIGHT$ (A$,4) da lugar a "GHIJ"
sintaxis: LEFT$ (cadena, número)
RIGHT$ (cadena, número)

```
10 REM ************************
15 REM * JUEGO DE CARACTERES *
20 REM *                      *
25 REM *     PROGRAMA 1       *
30 REM ************************
35 REM
40 PRINT"JUEGO DE CARACTERES":PRINT
45 FORX=33TO129
50 PRINT CHR$(X);
55 NEXTX
60 FORX=161 TO255
65 PRINT CHR$(X);
70 NEXTX
```

MID$** es una función más potente que puede utilizarse para obtener segmentos del interior de una cadena, y no como **LEFT$ o **RIGHT$**, que están limitadas a comenzar por uno de los extremos de la cadena.
sintaxis: MID$ (cadena, número 1, número 2)

MID$ (A$,N,M) extrae M caracteres comenzando en el N-simo carácter de la cadena A$. Si se omite el último argumento M, entonces se extraen todos los caracteres siguientes al N-simo.

CUATRO LETRAS

El programa 2 ilustra el troceado de cadenas mediante el uso de la función **MID$**. Después de introducir una frase el programa la presentará en pantalla con todas las palabras de cuatro letras sustituidas por asteriscos.

En la línea 50 se construye una cadena de un máximo de 80 caracteres, con la sentencia INPUT para los caracteres interiores, añadiendo un primer y un último carácter en blanco.

Se utiliza el puntero X para moverse de un carácter a otro. El programa verifica si el carácter X es un espacio, después si los cuatro caracteres siguientes son todos caracteres y, por último, si el siguiente carácter es otro espacio. Si se cumplen todas estas condiciones significa que ha encontrado una palabra de cuatro letras, que debe ser sustituida por cuatro asteriscos. Si alguna de las condiciones no se cumple, X se incrementa en una unidad y se procede a examinar el siguiente conjunto de seis caracteres.

```
10 REM ****************
15 REM * CUATRO LETRAS *
20 REM *              *
25 REM *  PROGRAMA 2   *
30 REM ****************
35 REM
40 PRINT:PRINT:PRINT"]CUATRO LETRAS"
45 PRINT:PRINT"ESCRIBA UNA FRASE"
50 INPUT A$:A$=" "+A$+" "
55 L=LEN(A$)
60 FORX=1TO L-5
65 IF MID$(A$,X,1)<> " " THEN95
70 FOR Y=1TO4
75 IF MID$(A$,X+Y,1)=" "THEN95
80 NEXT Y
85 IF MID$(A$,X+5,1)<>" " THEN95
90 A$=LEFT$(A$,X)+"****"+RIGHT$(A$,L-(X+4))
95 NEXTX
100 PRINT:PRINTA$
105 PRINT:PRINT"PULSE ESPACIO PARA REPETIR"
110 GETK$:IFK$<>" " THEN110
115 GOTO40
```

BUSQUEDA DE PALABRAS

Podemos utilizar las funciones de manejo de cadenas para identificar letras o palabras específicas en una frase. El programa 3 es un ejemplo. Al darle una frase y una palabra clave, lleva a cabo una búsqueda para ver si la palabra clave está en la frase.

```
10 REM ***********************
15 REM * BUSQUEDA DE PALABRAS *
20 REM *                     *
25 REM *      PROGRAMA 3      *
30 REM ***********************
35 REM
40 PRINT:PRINT"]BUSQUEDA DE PALABRAS"
45 PRINT:PRINT"ESC<IBA LA PALABRA CLAVE "
50 INPUT W$
55 PRINT:PRINT"ESCRIBA LA FRASE "
60 INPUT S$:S$=S$+" "
65 L=LEN(S$):P=1
70 FOR X=1TO L
75 IF MID$(S$,X,1)<>" " THEN 95
80 IF MID$(S$,P,X-P)<>W$ THEN P=X+1:GOTO 95
85 PRINT:PRINT"LA PALABRA ESTA EN LA FRASE"
90 GOTO105
95 NEXTX
100 PRINT:PRINT"LA PALABRA NO ESTA EN LA FRASE"
105 PRINT:PRINT"PULSE:"
110 PRINT"        ESPACIO PARA REPETIR"
115 PRINT"        OTRA TECLA PARA NUEVA CLAVE"
120 GET K$:IFK$=""THEN120
125 IF K$=" "THEN 55
130 GOTO 40
```

El programa busca espacios o caracteres en blanco, ya que al encontrar uno sabe que los caracteres que le preceden, hasta el espacio anterior, forman una palabra completa. Una vez que ha localizado una palabra lleva a cabo una comparación, para saber si es o no la palabra clave.

```
10 REM **************
15 REM *  MAANARAG   *
20 REM *             *
25 REM * PROGRAMA 5 *
30 REM **************
35 REM
40 PRINT:PRINT"♥ANAGRAMA"
45 PRINT:PRINT"DESENREDA LA PALABRA:  "
50 A$="":G$="":RESTORE
55 N=80:REM**NUMERO DE PALABRAS**
60 FORJ=1TO (RND(0)*N+1)
65 READW$
70 NEXT J
75 L=LEN(W$):WW$=W$
80 FORJ=1TOL
85 N=INT(1+RND(0)*L)
90 L$=MID$(W$,N,1):IFL$="*"THEN85
95 W$=LEFT$(W$,N-1)+"*"+MID$(W$,N+1)
100 A$=A$+L$
105 NEXTJ
110 IF A$=WW$ THEN 60
115 PRINT:PRINT,,A$
120 PRINT:INPUT"QUE PALABRA ES?   ";G$
125 IFG$=""THEN120
130 IF MID$(G$,1,1)="?"THEN PRINT:PRINT,"RESPUESTA ",WW$:GOTO145
135 IF WW$<>G$THEN PRINT:PRINT"PRUEBA OTRA VEZ":GOTO120
140 PRINT:PRINT"MUY BIEN"
145 PRINT:PRINT"PULSA ESPACIO PARA SEGUIR"
150 GET K$:IF K$=" "THEN 40
155 GOTO150
1000 REM      PALABRAS
1005 REM      ========
1010 DATA HIDROGENO,MONOPOLY,LOOPING,CHIMENEA,PAQUETE,CORAL,CRONICA,DUO,ELIPSE
1015 DATA FASCISMO,MIEL,CRUEL,MALARIA,RACIONAL,RACION,FRIO,LLORAR,COAXIAL,NIVEL
1020 DATADATO,EGOISMO,SANTO,BAUTIZO,ABACO,EON,TRIPAS,VIDA,VINO,PINO,RODILLA
1025 DATAGORRO,RANURA,MEMORIA,SERVICIO,OMEGA,SOCIAL,OLVIDAR,HOBY,RAQUETA,PERRO
1030 DATACOHETE,GARANTIA,COLUMNA,CABALLERO,GRANDE,RECIBE,SASTRE,TIO,QUIEN,SALIDA
1035 DATAGIRAR,ACIDO,PODER,ASOMAR,GERUNDIO,JOYA,CRUDO,CARNAVAL,BIQUINI,ISLA
1040 DATAJUNGLA,ZORRO,KAKI,ADQUIRIR,JUBILACION,ENIGMA,LIO,SALTO,ORACULO,TETERA
1045 DATAORTODOXO,INTERNO,MATAR,FALTA,IRONIA,FLEXIBLE,TERRORIFICO,MICRO,JUPITER
1050 REM   ETC.
```

¿Cómo modificaria usted el programa para que cuente el número de veces que aparece la palabra clave?

En el ejemplo anterior, el empleo de la sentencia INPUT supone una limitación en la longitud máxima de la frase, que no puede ser mayor de 80 caracteres. El ejemplo que sigue muestra de nuevo la técnica de búsqueda de palabras. Se trata de localizar una palabra clave entre una serie de palabras contenidas en sentencias DATA. Además de encontrar la palabra clave, el programa 4 muestra palabras que están relacionadas con dicha clave.

El programa se supone desarrollado por una empresa de *software* que cuenta con gran número de empleados, cada uno con distintos conocimientos y experiencia en diferentes máquinas y lenguajes de programación. El director necesitaba de alguna forma llevar un registro de la experiencia de cada empleado y algún método para obtener los nombres de todos los empleados con experiencia en una determinada materia. En cierto modo, necesitaba un sistema sencillo de gestión de base de datos.

El nombre de cada empleado se guarda en una sentencia DATA, seguido de sus conocimientos. Para distinguir entre el nombre y los conocimientos, el nombre va precedido del símbolo "£". El programa realiza una búsqueda elemento a elemento. Si la palabra va precedida de "£", el nombre del empleado se almacena temporalmente mientras se examinan sus conocimientos. Si posee los conocimientos requeridos, entonces se muestra su nombre en la pantalla. El programa continúa examinando el resto de las sentencias DATA hasta llegar al símbolo %, que indica el fin del archivo.

Esta es probablemente una de las

formas más simples que puede tomar un programa de gestión de base de datos, estando limitado solamente por el tamaño de la memoria del ordenador. Tiene la ventaja de que el director puede modificar fácilmente los registros de los empleados, sin más que añadir o eliminar elementos en las sentencias DATA. Partiendo de este ejemplo, usted puede modificar las sentencias DATA para construir una base de datos más acorde a sus necesidades.

El último programa ilustra sobre como manejar caracteres en un texto. Se elige una palabra aleatoriamente entre las contenidas en las sentencias DATA y luego se entremezclan sus letras para producir un anagrama. Este anagrama se muestra en pantalla al jugador, que debe intentar descubrir la palabra original. Escribiendo "?" el jugador puede ver cual es la palabra oculta. La lista de palabras puede hacerse mayor si se desea, añadiendo nuevas sentencias DATA con más palabras y modificando el valor de N en la línea 55 y que representa el número total de palabras.

```
10 REM ****************
15 REM * BASE DE DATOS *
20 REM *              *
25 REM *  PROGRAMA 4   *
30 REM ****************
35 REM
40 PRINT:PRINT"]SOFTWARE JUAN S.A."
45 RESTORE:N$="":S$=""
50 PRINT:INPUT"ESCRIBA EXPERIENCIA ";S$
55 PRINT
60 READ W$
65 IF MID$(W$,1,1)="%" THEN 85
70 IF MID$(W$,1,1)="£" THEN N$=MID$(W$,2):GOTO60
75 IF W$=S$ THEN PRINT W$;"        ";N$
80 GOTO 60
85 PRINT:PRINT"PULSA ESPACIO PARA SEGUIR"
90 GET K$:IF K$=" " THEN 40
95 GOTO90
1000 REM LINEAS DATA
1010 REM ===========
1015 DATA £GOMEZ J,FORTRAN,ADA,VAX,BASIC
1020 DATA £SMITH R,COBOL,APPLE,VAX,ATARI
1025 DATA £MARCOS L, BASIC,COBOL,PDP11,ADA,FORTRAN
1030 DATA £SINCLAIR M,BASIC,PASCAL,QL,VAX,COBOL,ALGOL
1035 DATA £DURAN J,UNIX,CP/M,PASCAL,VAX,APFLE
1040 DATA £HERRANZ A,IBM370,COBOL
1045 DATA £POZO D,BASIC,Z80,8064
2000 DATA %
```

Bingo

Si quieres organizar tu propio bingo sólo tienes que preparar los cartones, que todo el trabajo de ir sacando bolas lo realiza tu ordenador **VIC**, si le cargas este programa. **Bingo** ha sido escrito por **Angel León Negrín** y ha llegado a nuestra redacción desde Santa Cruz de Tenerife.

Cuando éste todo preparado, sólo hay que escribir RUN y empezarán a aparecer bolas al azar, en caracteres gigantes, sobre la pantalla. La aparición de la bola va acompañada de sonido, y después de un tiempo, suficiente para anotar el número en el cartón, aparece la siguiente bola. Al cantar un bingo, hay que pulsar cualquier tecla con lo que podrá verse en la pantalla una tabla de los números que han ido saliendo. Si el bingo cantado es correcto, se puede empezar de nuevo desde el principio, y si no lo es, el programa sigue sacando bolas desde donde fue parado.

El precio del cartón y el reparto de los premios corren de tu cuenta.




```
10 DIM T$(90),N(90),B$(1)
15 B$(0)=" ":B$(1)="●":G=32768
20 GOTO180
25 GETG$:IFG$<>""THEN300
30 BO=INT(RND(0)*90+1)
40 IFN(BO)<>0THEN30
50 N(BO)=BO
60 GOSUB 100
70 FORX=1TO2500:NEXT
80 GOTO25
100 D1=INT(BO/10):U1=BO-10*D1
105 DC=48+D1::UN=48+U1
110 IFDC=48THENDC=32
120 POKE36865,170
130 PRINT"▮▮▮▮▮▮▮▮"
140 R=DC:SP$="▮▮▮":GOSUB170
150 R=UN:PRINT"▮▮▮▮▮▮▮▮▮":SP$="▮▮▮▮▮▮▮▮▮▮▮▮":GOSUB170
155 POKE36878,15
160 FORZ=170TO38STEP-12
161 POKE36865,Z
162 POKE36876,279-Z:NEXT
163 FORT=1TO20:NEXT
164 POKE36878,0:POKE36876,0
165 RETURN
170 FORL=0TO7
171 A=PEEK(G+8*R+L):A$=""
172 FORB=1TO8
173 C=INT(A/2):D=A-C*2
174 A$=B$(D)+A$
175 A=C:NEXT
176 PRINTSP$;A$:NEXT
177 RETURN
180 FORJ=0TO90:READT$(J):NEXT
190 POKE36879,110:PRINTCHR$(5)
200 PRINT"▮      *** BINGO ***"
210 PRINT"▮▮▮AL CANTAR BINGO,"
220 PRINT:PRINT" PULSA CUALQUIER TECLA"
230 PRINT" TE MOSTRARA LAS BOLAS"
240 PRINT" QUE HAN SALIDO"
250 PRINT"▮▮▮* PULSA UNA TECLA"
260 PRINT:PRINT"  PARA EMPEZAR"
270 GETG$:IFG$=""THEN270
280 GOTO30
300 PRINT"▮"
310 FORX=1TO81STEP10
```



```
320 FORI=XTOX+9
330 J=N(I)
340 PRINTT$(J);
350 NEXT
360 PRINT:PRINT
370 NEXT
380 PRINT"■■BINGO CORRECTO?"
390 PRINT"■ ■S■ SI  ■N■ NO"
410 GETG$:IFG$="S"THENRUN
420 IFG$="N"THEN30
430 GOTO410
500 DATA"__"," 1","■ 2■"," 3","■ 4■"," 5","■ 6■"," 7","■ 8■"," 9","■10■"
510 DATA"■11■","12","■13■","14","■15■","16","■17■","18","■19■","20
520 DATA"21","■22■","23","■24■","25","■26■","27","■28■","29","■30■"
530 DATA"■31■","32","■33■","34","■35■","36","■37■","38","■39■","40"
540 DATA"41","■42■","43","■44■","45","■46■","47","■48■","49","■50■"
550 DATA"■51■","52","■53■","54","■55■","56","■57■","58","■59■","60"
```


cuantas sentencias DA-
que recomendamos a
quieran copiarlo, que
cia y anden con cuida-
scape ningún número o
También, antes de es-
ne se aseguran de haber
programa en *cassette,*
error hace necesario
enador.
nstrucciones necesarias
gar van incluidas en el
ma.

REMIADO CON .000 PESETAS

```
140 FOR S=16320 TO 16382:READ Q(7):        POKES,Q(7):NEXT
150 V=53248
160 POKE V+21,255
170 POKE V+0,132:POKEV+2,172:POKEV+4,212:  POKE V+1,99 :POKEV+3,99 :POKEV+5,99
180 PRINT:PRINT:PRINT:PRINT:PRINTW$
190 PRINT"■■■■■■■■■■■■■■■■■■■"SPC(10)"* F1: PARAR IZQUIERDA"
200 PRINT"■■"SPC(10)"* F3: PARAR CENTRO"
210 PRINT"■■"SPC(10)"* F5: PARAR DERECHA"
220 POKE V+0,132:POKEV+2,172:POKEV+4,212:  POKE V+1,99 :POKEV+3,99 :POKEV+5,99
```

```
320 FORI=XTOX+9
330 J=N(I)
340 PRINTT$(J);
350 NEXT
360 PRINT:PRINT
370 NEXT
380 PRINT"▒▒BINGO CORRECTO?"
390 PRINT"▒ ▒S■ SI   ▒N■ NO"
410 GETG$:IFG$="S"THENRUN
420 IFG$="N"THEN30
430 GOTO410
500 DATA"__"," 1","▒ 2■"," 3","▒ 4■"," 5","▒ 6■"," 7","▒ 8■"," 9","▒10■"
510 DATA"▒11■","12","▒13■","14","▒15■","16","▒17■","18","▒19■","20
520 DATA"21","▒22■","23","▒24■","25","▒26■","27","▒28■","29","▒30■"
530 DATA"▒31■","32","▒33■","34","▒35■","36","▒37■","38","▒39■","40"
540 DATA"41","▒42■","43","▒44■","45","▒46■","47","▒48■","49","▒50■"
550 DATA"▒51■","52","▒53■","54","▒55■","56","▒57■","58","▒59■","60"
560 DATA"61","▒62■","63","▒64■","65","▒66■","67","▒68■","69","▒70■"
570 DATA"▒71■","72","▒73■","74","▒75■","76","▒77■","78","▒79■","80"
580 DATA"81","▒82■","83","▒84■","85","▒86■","87","▒88■","89","▒90■"
```

Tragaperras

CBM 64

Para los viciosos de las máquinas tragaperras (éstas de frutas que se encuentran casi en cualquier sitio), nada mejor que este programa que nos envía **Antonio Carvajal de Madrid.** Es una simulación casi exacta de una de tales máquinas de frutas y, a decir verdad, lo único que se echa en falta es una ranura por la que salgan las monedas de los premios conseguidos. El programa hace un abundante y muy adecuado uso de los Sprites, además de aprovechar las capacidades musicales del **64** haciendo que a los diferentes premios correspondan tonadillas diferentes, todas ellas muy conocidas.

El listado del programa es largo y contiene unas cuantas sentencias DATA, por lo que recomendamos a aquéllos que quieran copiarlo, que tengan paciencia y anden con cuidado, no se les escape ningún número o alguna coma. También, antes de escribir RUN, que se aseguran de haber guardado el programa en *cassette,* por si algún error hace necesario apagar el ordenador.

Todas las instrucciones necesarias para poder jugar van incluidas en el propio programa.

PREMIADO CON 5.000 PESETAS

```
10 PRINT"♥"
20 T$="                ┌─────┬─────┬─────┐                 "
30 U$="                │     │     │     │                 "
40 V$="                └─────┴─────┴─────┘                 "
50 W$=T$+U$+U$+U$+V$
60 B$="                                                      "
70 POKE 53281,1
80 FOR S=15936 TO 15998:READ Q(1):                  POKES,Q(1):NEXT
90 FOR S=16000 TO 16062:READ Q(2):                  POKES,Q(2):NEXT
100 FOR S=16064 TO 16126:READ Q(3):                  POKES,Q(3):NEXT
110 FOR S=16128 TO 16190:READ Q(4):                  POKES,Q(4):NEXT
120 FOR S=16192 TO 16254:READ Q(5):                  POKES,Q(5):NEXT
130 FOR S=16256 TO 16318:READ Q(6):                  POKES,Q(6):NEXT
140 FOR S=16320 TO 16382:READ Q(7):                  POKES,Q(7):NEXT
150 V=53248
160 POKE V+21,255
170 POKE V+0,132:POKEV+2,172:POKEV+4,212:    POKE V+1,99 :POKEV+3,99 :POKEV+5,99
180 PRINT:PRINT:PRINT:PRINT:PRINTW$
190 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"SPC(10)"* F1: PARAR IZQUIERDA"
200 PRINT"▒▒"SPC(10)"* F3: PARAR CENTRO"
210 PRINT"▒▒"SPC(10)"* F5: PARAR DERECHA"
220 POKE V+0,132:POKEV+2,172:POKEV+4,212:    POKE V+1,99 :POKEV+3,99 :POKEV+5,99
```

Concurso

Viene de la página anterior

```
230 S=54272
240 FOR L=S TO S+24:POKE L,0:NEXT
250 POKE S+5,15:POKES+6,129:POKE S+24,15
260 FOR I=1 TO 8:READ AV(I),BV(I),DV(I):    NEXT I
270 FOR I=1 TO 7:READ AP(I),BP(I),DP(I):    NEXT I
280 FOR I=1 TO 6:READ AR(I),BR(I),DR(I):    NEXT I
290 FOR I=1 TO 6:READ AH(I),BH(I),DH(I):    NEXT I
300 FOR I=1 TO 7:READ AE(I),BE(I),DE(I):    NEXT I
310 FOR I=1 TO 8:READ AS(I),BS(I),DS(I):    NEXT I
320 FOR I=1 TO 6:READ AA(I),BA(I),DA(I):    NEXT I
330 FOR I=1 TO 9:READ AM(I),BM(I),DM(I):    NEXT I
340 FOR I=1 TO 8:READ AF(I),BF(I),DF(I):    NEXT I
350 FOR I=1 TO10:READ AQ(I),BQ(I),DQ(I):    NEXT I
360 FOR I=1 TO 9:READ AI(I),BI(I),DI(I):    NEXT I
370 FOR I=1 TO 3:READ AC(I),BC(I),DC(I):    NEXT I
380 GOTO 960
390 POKE 198,0
400 FORL=1 TO 40
410 GET A$
420 IF X<>0 THEN 530
430 IF A$="■" THEN X=1:GOTO 530
440 P0=INT(RND(0)*7)+249:POKE 2040,P0
450 IF P0=249 THEN C0=4:GOTO 520
460 IF P0=250 THEN C0=14:GOTO 520
470 IF P0=251 THEN C0=8:GOTO 520
480 IF P0=252 THEN C0=2:GOTO 520
490 IF P0=253 THEN C0=7:GOTO 520
500 IF P0=254 THEN C0=5:GOTO 520
510 IF P0=255 THEN C0=6
520 POKE V+39,C0
530 IF Y<>0 THEN 640
540 IF A$="■" THEN Y=1:GOTO 640
550 P1=INT(RND(0)*7)+249:POKE 2041,P1
560 IF P1=249 THEN C1=4:GOTO 630
570 IF P1=250 THEN C1=14:GOTO 630
580 IF P1=251 THEN C1=8:GOTO 630
590 IF P1=252 THEN C1=2:GOTO 630
600 IF P1=253 THEN C1=7:GOTO 630
610 IF P1=254 THEN C1=5:GOTO 630
620 IF P1=255 THEN C1=6
630 POKE V+40,C1
640 IF Z<>0 THEN 750
650 IF A$="■" THEN Z=1:GOTO 750
660 P2=INT(RND(0)*7)+249:POKE 2042,P2
670 IF P2=249 THEN C2=4:GOTO 740
680 IF P2=250 THEN C2=14:GOTO 740
690 IF P2=251 THEN C2=8:GOTO 740
700 IF P2=252 THEN C2=2:GOTO 740
710 IF P2=253 THEN C2=7:GOTO 740
720 IF P2=254 THEN C2=5:GOTO 740
730 IF P2=255 THEN C2=6
740 POKE V+41,C2
750 NEXT L
760 D=D+25
770 IF P0=249AND P1=249AND P2=249 THEN G=  75:GOTO 2600
780 IF(P0=249AND P1=249)OR(P1=249 AND P2=  249) THEN G=50:GOTO 2670
790 IF P0=249ORP2=249 THENG=25:GOTO 2740
```



```
800 IF P0=255 AND P1=250 AND P2=250 THEN  G=100:GOTO 2530
810 IF P0=250 AND P1=250 AND P2=255 THEN  G=100:GOTO 2530
820 IF P0=250 AND P1=250 AND P2=250 THEN  G=125:GOTO 2460
830 IF P0=255 AND P1=251 AND P2=251 THEN  G=150:GOTO 2390
840 IF P0=251 AND P1=251 AND P2=255 THEN  G=150:GOTO 2390
850 IF P0=251 AND P1=251 AND P2=251 THEN  G=200:GOTO 2320
860 IF P0=255 AND P1=252 AND P2=252 THEN  G=250:GOTO 2250
870 IF P0=252 AND P1=252 AND P2=255 THEN  G=250:GOTO 2250
880 IF P0=252 AND P1=252 AND P2=252 THEN  G=300:GOTO 2180
890 IF P0=255 AND P1=253 AND P2=253 THEN  G=350:GOTO 2110
900 IF P0=253 AND P1=253 AND P2=255 THEN  G=350:GOTO 2110
910 IF P0=253 AND P1=253 AND P2=253 THEN  G=400:GOTO 2040
920 IF P0=255 AND P1=254 AND P2=254 THEN  G=450:GOTO 1970
930 IF P0=254 AND P1=254 AND P2=255 THEN  G=450:GOTO 1970
940 IF P0=254 AND P1=254 AND P2=254 THEN  G=500:GOTO 1830
950 IF P0=255 AND P1=255 AND P2=255 THEN  G=5000:GOTO 1900
960 PRINT"■":PRINT"PULSE UNA TECLA PARA JUGAR"
970 PRINT"■DINERO GASTADO =";" ";D:PRINT"DINERO GANADO  =";" ";GA:SA=GA-D
980 SA$=STR$(SA):SA$=SA$+"     ":PRINT"SALDO";TAB(15);"=";TAB(17);SA$
990 GETP$:IFP$=""THEN 990
1000 X=0:Y=0:Z=0:IFP$<>""THEN 390
1010 GA=GA+G:PRINT"■":PRINTTAB(240):PRINTTAB(240):PRINT"■PREMIO =";G;"■ PESETAS"
1020 PRINT"TT":FOR E=1TO1500:NEXT:PRINTB$:GOTO 960
1030
```

Concurso

Viene de la página anterior

```
1040 REM ***** CEREZA ******************
1050 :
1060 DATA 28,0,0,14,1,248,3,7,254,1,143,255,0,223,254,0,115,248,0,48,0,0,48,0
1070 DATA 0,24,0,7,219,224,15,255,240,28,126,56,24,60,24,24,60,24,28,126,56
1080 DATA 15,231,240,7,195,224,0,195,0,0,195,0,0,126,0,0,60,0
1090 :
1100 REM ***** DATIL ******************
1110 :
1120 DATA0,0,28,0,0,56,0,0,112,0,0,224,0,63,192,0,255,128,3,255,128,15,255,128
1130 DATA 31,255,128,63,255,128,63,255,128,127,255,0,127,255,0,255,254,0
1140 DATA 255,254,0,255,252,0,255,248,0,255,224,0,127,192,0,63,0,0,0,0,0
1150 :
1160 REM ***** NARANJA ****************
1170 :
1180 DATA 0,0,0,0,0,0,1,255,128,7,195,224,30,195,120,54,255,124,127,0,238
1190 DATA 111,255,254,251,123,219,223,239,127,246,251,219,223,223,127
1200 DATA 245,187,219,95,238,254,123,123,214,63,238,252,29,187,216,7,239,224
1210 DATA 1,255,128,0,0,0,0,0,0
1220 :
1230 REM ***** FRESA ******************
1240 :
1250 DATA 0,6,0,0,12,0,0,24,0,15,153,240,31,219,248,60 254,124,127,255,254
1260 DATA 243,249,243,255,255,255,231,159,63,255,255,255,126,115,158,63,255,252
1270 DATA 28,252,248,15,255,240,7,231,224,3,255,192,1,255,128,0,231,0,0,126,0
1280 DATA 0,60,0
1290 :
1300 REM ***** CAMPANA ****************
1310 :
1320 DATA 0,60,0,0,126,0,0,219,0,0,255,0,0,255,0,0,255,0,1,255,128,1,255,128
1330 DATA 3,90,192,2,255,64,7,255,224,29,90,184,55,255,236,111,255,246
1340 DATA 120,24,30,224,24,7,192,24,3,192,60,3,96,60,6,60,0,60,7,255,224
1350 :
1360 REM ****** SANDIA ****************
1370 :
1380 DATA 0,3,224,0,63,240,1,255,152,7,255,12,15,254,198,31,254,198,63,254,6
1390 DATA 127,252,51,127,252,51,255,253,131,255,253,155,255,252,27,127,253,131
1400 DATA 127,253,179,63,254,54,31,254,6,15,254,102,7,255,108,1,255,152
1410 DATA 0,63,240,0,3,224
1420 :
1430 REM ***** GANCHO ****************
1440 :
1450 DATA 127,240,0,127,240,0,112,112,0,112,112,0,112,0,0,112,0,0,112,127,224
1460 DATA 112,127,224,112,112,0,112,112,0,112,112,0,127,255,128,127,255,128
1470 DATA 0,112,0,0,112,0,0,112,0,0,112,0,0,112,0,0,112,0,0,112,0,0,112,0
1480 REM ++++++ VACA ++++++++++++++++++
1490 DATA 21,154,200,24,63,50,21,154,50,19,63,200,21,154,50,19,163,50
1500 DATA 18,42,200,14,107,200
1510 REM ++++++ PAJARITOS +++++++++++++
1520 DATA 21,154,100,21,154,100,24,63,100,24,63,50,18,42,100,18,42,100
1530 DATA 21,154,100
1540 REM ++++++ RICO ++++++++++++++++++
1550 DATA 21,154,200,19,63,100,21,154,200,19,63,100,18,42,200,14,107,200
1560 REM ++++++ HABIA UNA VEZ +++++++++
1570 DATA 16,47,50,21,154,100,21,154,70,24,63,100,24,63,50,27,56,200
1580 REM ++++++ EL PATIO ++++++++++++++
1590 DATA 16,47,100,21,154,100,21,154,100,21,154,100,27,56,150,21,154,250
1600 DATA 21,154,100
1610 REM ++++++ SI YO TUVIERA +++++++++
```

```
1620 DATA21,154,90,21,154,90,21,154,90,22,227,90,22,227,90,22,227,90,25,177,200
1630 DATA 25,177,100
1640 REM ++++++ AL PASAR ++++++++++++++
1650 DATA 21,154,100,22,227,100,25,177,250,34,75,250,32,94,100,32,94,100
1660 REM ++++++ MENTIRAS ++++++++++++++
1670 DATA 14,107,90,18,42,90,21,154,250,22,227,90,25,177,250,22,227,50
1680 DATA 21,154,50,19,63,50,21,154,150
1690 REM ++++++ FREDEJACQUE +++++++++++
1700 DATA 21,154,125,24,63,100,27,56,100,21,154,125
1710 DATA 21,154,100,24,63,100,27,56,100,21,154,200
1720 REM ++++++ QUINTO LEVANTA ++++++++
1730 DATA 21,154,150,21,154,150,27,56,70,21,154,150,21,154,150,27,56,70
1740 DATA 21,154,70,27,56,70,21,154,70,16,47,200
1750 REM ++++++ INTERNACIONAL +++++++++
1760 DATA 16,47,100,16,47,70,21,154,200,21,154,200,24,63,200,24,63,250
1770 DATA 32,94,350,27,56,50,21,154,50
1780 REM ++++++ CINCO DUROS +++++++++++
1790 DATA 51,97,100,51,97,100,51,97,100
1800
1810 REM ***** MUSICA ****************
1820
1830 FOR M=1 TO 8
1840 POKE S+1,AV(M):POKE S,BV(M)
1850 POKE S+4,33
```

Concurso

Viene de la página anterior

```
1860 FOR T=1 TO DV(M):NEXT
1870 POKE S+4,32 :FOR T=1 TO 50:NEXT
1880 NEXT M
1890 GOTO 1010
1900 FOR M1=1 TO 2:FOR M=1 TO 7
1910 POKE S+1,AP(M):POKE S,BP(M)
1920 POKE S+4,33
1930 FOR T=1 TO DP(M):NEXT
1940 POKE S+4,32:FOR T=1 TO 50:NEXT
1950 NEXT M:FORM2=1 TO 50:NEXTM2,M1
1960 GOTO 1010
1970 FOR M=1 TO 6
1980 POKE S+1,AR(M):POKE S,BR(M)
1990 POKE S+4,33
2000 FOR T=1 TO DR(M):NEXT
2010 POKE S+4,32:FOR T=1 TO 50:NEXT
2020 NEXT M
2030 GOTO 1010
2040 FOR M=1 TO 6
2050 POKE S+1,AH(M):POKE S,BH(M)
2060 POKE S+4,33
2070 FOR T=1 TO DH(M):NEXT
2080 POKE S+4,32:FOR T=1 TO 50:NEXT
2090 NEXT M
2100 GOTO 1010
2110 FOR M=1 TO 7
2120 POKE S+1,AE(M):POKE S,BE(M)
2130 POKE S+4,33
2140 FOR T=1 TO DE(M):NEXT
2150 POKE S+4,32:FOR T=1 TO 50:NEXT
2160 NEXT M
2170 GOTO 1010
2180 FOR M=1 TO 8
2190 POKE S+1,AS(M):POKE S,BS(M)
2200 POKE S+4,33
2210 FOR T=1 TO DS(M):NEXT
2220 POKE S+4,32:FOR T=1 TO 50:NEXT
2230 NEXT M
2240 GOTO 1010
2250 FOR M=1 TO 6
2260 POKE S+1,AA(M):POKE S,BA(M)
2270 POKE S+4,33
2280 FOR T=1 TO DA(M):NEXT
2290 POKE S+4,32:FOR T=1 TO 50:NEXT
2300 NEXT M
2310 GOTO 1010
2320 FOR M=1 TO 9
2330 POKE S+1,AM(M):POKE S,BM(M)
2340 POKE S+4,33
2350 FOR T=1 TO DM(M):NEXT
2360 POKE S+4,32:FOR T=1 TO 50:NEXT
2370 NEXT M
2380 GOTO 1010
2390 FOR M=1 TO 8
2400 POKE S+1,AF(M):POKE S,BF(M)
2410 POKE S+4,33
2420 FOR T=1 TO DF(M):NEXT
```

```
2430 POKE S+4,32:FOR T=1 TO 50:NEXT
2440 NEXT M
2450 GOTO 1010
2460 FOR M=1 TO 10
2470 POKE S+1,AQ(M):POKE S,BQ(M)
2480 POKE S+4,33
2490 FOR T=1 TO DQ(M):NEXT
2500 POKE S+4,32:FOR T=1 TO 50:NEXT
2510 NEXT M
2520 GOTO 1010
2530 FOR M=1 TO 9
2540 POKE S+1,AI(M):POKE S,BI(M)
2550 POKE S+4,33
2560 FOR T=1 TO DI(M):NEXT
2570 POKE S+4,32:FOR T=1 TO 50:NEXT
2580 NEXT M
2590 GOTO 1010
2600 FOR M=1 TO 3
2610 POKE S+1,AC(M):POKE S,BC(M)
2620 POKE S+4,33
2630 FOR T=1 TO DC(M):NEXT
2640 POKE S+4,32:FOR T=1 TO 50:NEXT
2650 NEXT M
2660 GOTO 1010
```

Concurso

Viene de la página anterior

```
2670 FOR M=1 TO 2
2680 POKE S+1,AC(M):POKE S,BC(M)
2690 POKE S+4,33
2700 FOR T=1 TO DC(M):NEXT
2710 POKE S+4,32:FOR T=1 TO 50:NEXT
2720 NEXT M
2730 GOTO 1010
2740 FOR M=1 TO 1
2750 POKE S+1,AC(M):POKE S,BC(M)
2760 POKE S+4,33
2770 FOR T=1 TO 200:NEXT
2780 POKE S+4,32:FOR T=1 TO 50:NEXT
2790 NEXT M
2800 GOTO 1010
```



Socarrado

VIC-20

Socarrado es el nombre del muñeco protagonista de este juego para el **VIC 20,** que ha sido creado por **José Fernando Ferrando** y que llega desde Valencia. La misión de este personaje, que el jugador controla desde el teclado, es recoger unas bolsas de dinero, que están en la parte inferior derecha de la pantalla, y llevarlas hasta la parte inferior izquierda donde se van apilando unas encima de otras. La dificultad está en que durante el trayecto, vay cayendo, como si lloviera, unas piedras o bolas que acaban con el personaje si aciertan a caer sobre él. También a lo largo del trayecto hay tres muros de defensa, bajo los que puede cobijarse el personaje, pero que van siendo poco a poco destruidos por la lluvia de piedras. Por cada bolsa de dinero transportada a salvo, se rehacen parte de las defensas destruidas, y cuando el número de bolsas a salvo sea de 16, el jugador pasa a disponer de un personaje más, aparte de los cinco con que se comienza el juego.

```
10 PRINT"[illegible]":GOSUB30000:HT=0:XX=0:TE=0:SC=0:CA=0
20 VR=7680:YO=22:VV=38400:NI=5
25 Y$="[illegible]"
30 A1$="[illegible]"
40 A2$="[illegible]"
50 A3$="[illegible]"
60 A4$="[illegible]"
65 A5$="[illegible]"
66 A6$="[illegible]"
67 GOSUB6000:GOSUB8000
70 PRINT"[illegible]":FORI=1TO11:PRINT"[illegible]":NEXT
75 PRINT"                    "
100 FORI=0TO21:POKEVR+YO*22+I,160:POKEVV+YO*22+I,160:NEXT
200 POKE4,INT(RND(0)*5)+2
205 GOTO9000
210 A=PEEK(197):IFA=29ANDXX>1THENXX=XX-1
220 IFA=37ANDXX<17THENXX=XX+1
225 IFTE=1THEN260
227 IFTE=0ANDXX=17THENTE=1
230 IFXXAND1THEN250
240 PRINTY$SPC(XX)A1$:GOTO300
250 PRINTY$SPC(XX)A2$:GOTO300
260 IFTE=1ANDXX=1THENTE=0:GOSUB8000:GOSUB7000
265 IFXXAND1THEN275
270 PRINTY$SPC(XX)A3$:GOTO300
275 IFXX=17THEN290
```

```
280 PRINTY$SPC(XX)A4$:GOTO300
290 PRINTY$SPC(XX)LEFT$(A4$,LEN(A4$)-1)
300 GOTO200
2000 PRINT"]MMMM1 ** GAME OVER **"
2010 PRINT"MM TRY AGAIN (Y OR N) ?"
2020 PRINT"M SCORE ="SC
2030 IFHI<SCTHENHI=SC
2040 PRINT"M HI-SCORE ="HI
2050 A=PEEK(197):IFA<>11ANDA<>28THEN2050
2060 IFA=11THEN10
2070 PRINT"M END JOB":END
3000 ONINT(RND(0)*3)+1GOTO3010,3020,3030
3010 SP=2:GOTO3100
3020 SP=8:GOTO3100
3030 SP=14
3100 PRINT"SMMMMMM":FORI=1TOINT(RND(0)*8+3):PRINTSPC(SP)"R♦♦♦♦■":NEXT:RETURN
5000 PRINTY$SPC(XX)"]   "
5005 IFTE=1THEN5200
5100 FORI=1TO10:PRINTY$SPC(XX)A1$
5105 FORJ=1TO30:NEXT
5110 PRINTY$SPC(XX)A2$
5120 FORJ=1TO30:NEXT:NEXT:GOTO5300
5200 IFXX=17THENXX=16
5202 FORI=1TO10:PRINTY$SPC(XX)A3$
5205 FORJ=1TO30:NEXT
5210 PRINTY$SPC(XX)A4$
5220 FORJ=1TO30:NEXT:NEXT
5300 PRINTY$SPC(XX)A5$
5310 FORI=1TO2000:NEXT
5315 PRINTY$SPC(XX)A6$:XX=1:TE=0:GOSUB8000
5320 NI=NI-1:IFNI=0THENGOSUB6000:GOTO2000
5330 GOSUB6000:GOTO70
6000 PRINT"S RSCORE■"STR$(SC)TAB(15)STR$(NI)" RSFC■"
6010 RETURN
7000 SC=SC+INT(RND(0)*10)*100:GOSUB6000:GOSUB3000
7010 PRINT"SMMMMMMMMMMMMMMMMMMMMR";
7015 CA=CA+1:IFCA=16THENNI=NI+1:CA=0
7020 PRINTLEFT$("£■E■E■E■E■E■E■E■E■E■E■E■E■E■E■E",CA*3)
7030 RETURN
8000 PRINTY$"MM"SPC(20)"RE":RETURN
9000 RN=RND(0)+SC/3000+.4
9002 IFRN<1THEN9030
9005 FORI=1TORN
9010 PRINT"SM"SPC(INT(RND(0)*22))"*"
9020 NEXT
9030 SYS4096+2560
9100 IFPEEK(VR+YO*19+XX)=42THEN5000
9110 IFPEEK(VR+YO*18+XX+1)=42THEN5000
9120 IFPEEK(VR+YO*19+XX+2)=42THEN5000
9999 GOTO210
20000 DATA 169,0,133,0,169,32,133,1,160,0,169,0,133
20010 DATA 2,169,152,133,3,177,0,201,42,208,57,169,32
20020 DATA 145,0,24,165,0,105,22,133,5,165,1,105,0
20030 DATA 133,6,177,5,201,32,240,15,201,160,240,30,201
20040 DATA 156,240,26,169,32,145,5,76,81,26,169,42,145
20050 DATA 5,165,5,133,2,24,165,6,105,120,133,3,165
20060 DATA 4,145,2,136,208,190,198,1,198,3,165,3,165
20070 DATA 1,201,29,208,178,96,170,170,170,170,170,170,170
20080 DATA 170,170,170,170,170,170,170,170,170,170,170,170,170
30000 RESTORE:FORI=6656TO6752:READA:POKEI,A:NEXT:RETURN
```

PROGRAMA COMPROBADO commodore Magazine

VIC 20

Erratas

● Por un lamentable error, en el juego publicado en nuestro número 1, llamado **TRON,** no apareció la indicación correspondiente. El juego está destinado al **Vic 20** sin ampliar.

● Los duendes de las redacciones siempre hacen sus trastadas. En **Commodore Magazine** tampoco podía faltar el travieso Sprite. De todas formas su broma no le servirá de mucho. Simplemente con añadir este corto trozo de programa al titulado **"El dragón Temible",** publicado en el N.° 2, todo estará en orden.

● Hemos recibido algunas cartas y llamadas haciendo referencia al truco publicado en el N.° 3 de la revista, en la página 13, bajo el título de **"Así se congela la pantalla".**

Pues bien, el programa no es ni más ni menos que un cargador de lenguaje máquina desde BASIC. El bucle FOR-NEXT de la línea 20 es un contador, que altera los contenidos de las posiciones de memoria, desde la posición 49152 hasta la 49161. Cada vez que se incrementa el contador, se toma un dato de la sentencia DATA de la línea 30 y se deposita en la posición correspondiente. Hasta aquí todo correcto. Una vez que se ha tecleado el truco en el ordenador hay que escribir RUN y apretar la tecla Return. Así se altera el contenido de las posiciones que nos interesa. Una vez hecho esto, se teclea NEW (Return), con lo que borramos el programa en BASIC, pero no así las posiciones de memoria alteradas. Cada vez que tecleemos SHIFT, el truco hará su efecto.

Por ser un programa que afecta a posiciones de memoria RAM, habrá que cargarlo cada vez que se conecte.

```
808 GOTO800
810 IF PEEK(I)=35 ANDO=0 THEN 310
812 I=I-L
814 POKEQ+2,T+(O*5)
815 POKEI,R:POKEI+E,1:I=I-L:O=O-1:IFO=0ANDU=1 THEN POKED,R:POKED+E,1:U=0:GOTO185
816 IF O=0 THEN POKEQ+2,0:GOTO310
818 GOTO 814
900 POKEI,33:POKEI+E,7:FORM=180 TO 235 STEP2:POKEQ+2,M:NEXT
901 POKEQ+4,N:FORM=180TO235 STEP2:POKEQ+2,M:FORN=1TO10:NEXTN:NEXTM:POKEQ+2,0
902 POKEQ+4,0:W=W+(H-INT(TI/60)):H=H-5:IF H=5 THEN H=60
904 U=1:O=O+1:GOTO814
```

CONCURSO CALC RESULT

UN PREMIO DE 80.000 PTAS. EN MATERIAL COMMODORE

a elegir por el ganador, espera a la aplicación más original y útil que envien nuestros lectores.

Las hojas de trabajo ofrecen elevadas posibilidades de utilización. Sin embargo, creemos que en nuestro país todavía no se ha captado el enorme potencial que ofrecen. Por todo ello, "COMMODORE MAGAZINE" convoca un concurso de aplicaciones desarrolladas bajo CALC RESULT, sean financieras, dietéticas, etc...

■ El fallo del concurso se publicará en el número 4 de nuestra revista.

■ Para participar se enviará una detallada descripción de los objetivos que pretende la aplicación y la metodología utilizada.

■ El premio podrá ser declarado desierto, prorrogando en 2 meses la aceptación de nuevas aplicaciones en tal caso.

■ La aplicación premiada será publicada en forma de artículo. En caso de empate, el premio se dividirá en partes iguales entre los ganadores.

■ Los miembros del Jurado serán elegidos por "Commodore Magazine" entre cualificados profesionales que evaluarán la utilidad de la aplicación, siendo su decisión inapelable.

■ La fecha tope para la admisión de aplicaciones es el 1 de mayo de 1984. Todas las aplicaciones deben enviarse a:



Calc Result
"Commodore Magazine",
C/Bravo Murillo, 377. Madrid -20

¿Qué son los i...

En esta breve introducción intentaremos clarificar parte de la confusión y mística que rodea al campo de las comunicaciones de datos. Este es un mundo lleno de palabras, algunas tan inteligibles como los jeroglíficos —incluso para los entendidos en ordenadores.

Es de esperar que, de acuerdo con esto, los sortilegios extraños de "establezca la velocidad de los baudios y conecte a full duplex" puedan significar algo.

Existen tres "enchufes" comunes de comunicaciones de datos para la mayoría de los micros. Estos son: **RS232, IEEE 488** y **Centronics.** El único uso de éste último es controlar a las impresoras, y se le ha puesto nombre en honor de los fabricantes de impresoras que lo desarrollaron.

Los otros dos, **RS232** e **IEEE 488**, pueden comunicarse con una variedad de dispositivos, incluyendo las impresoras. Sin embargo, de los dos el **RS232** es, con mucho, el menos "estructurado" —esto es, el **RS232** no define el sistema total de comunicaciones, sino únicamente las patillas y señales que proporciona.

Es con ésta idea que esta introducción pretenderá dar una importancia mucho mayor a éste estandard, ya que es el que causa la mayoría de los problemas a los usuarios.

Antes que nada, debemos definir una serie de conceptos. La mayoría probablemente tenga conocimientos acerca de los bits y bytes. Pero vamos a refrescar vuestras memorias.

Un bit es un dígito binario 0 ó 1; en términos eléctricos, "off" u "on". Por su parte, un bit no constituye una parte importante de uso. Sin embargo, combinado en grupos de ocho (un byte) pueden ser utilizados para representar caracteres.

¿Por qué ocho?

Usted puede deducir de la tabla que presentamos abajo que para representar el alfabeto completo en mayúsculas y minúsculas, números y puntuación común, se requiere un sistema capaz de representar por lo menos 72 caracteres.

Esto necesita segmentos de 7 bits cada uno, permitiendo la posibilidad de 128 caracteres. El bit octavo —llamado el bit "ocho"— es utilizado para otros fines.

El código de 7 bits, que es utilizado para representación de caracteres, es aquel definido por el **ANSI**, el **American National Standards Institute**, y es conocido como el **ASCII** o **American Standard Code for Information Interchange.**

Letras mayúsculas	=	26
Letras minúsculas	=	26
Números	=	10
Puntuación	=	10
		72
4 bits (2^4)	=	16
5 bits (2^5)	=	32
6 bits (2^6)	=	64
7 bits (2^7)	=	128
8 bits (2^8)	=	256

Otro sistema de representación de caracteres en existencia es el **EBCDIC** (Extended Binary Coded Decimal Interchange Code - Código Ampliado de Caracteres Decimales Codificados en Binario para el Intercambio de la Información), pero no es de interés para la mayoría de los usuarios de micros, ya que principalmente sólo se le utiliza en ordenadores IBM más grandes.

Habiendo definido la representación de caracteres y bytes podemos pasar a describir las dos maneras básicas de enviar datos: en serie y en paralelo.

La transmisión en paralelo es el proceso de enviar un carácter íntegro de una vez, por medio de un grupo de cables en paralelo, uno para cada uno de los bits (más algunos cables de control).

La transmisión en serie, por otro lado, envía un bit por vez y requiere por lo menos siete transmisiones para un carácter.

Tanto el **IEEE 488** como el **Centronics** son interfaces de transmisión en paralelo, mientras que el **RS232** es un interface serie. La velocidad máxima teórica de la transmisión de datos es más elevada en las transmisiones en paralelo que en las transmisiones en serie, pero ésta última está mejor adaptada para las distancias más largas.

La transmisión en paralelo no es tan satisfactoria por una serie de razones. Un factor obvio seríe el costo de todos los cables extra requeridos. Además, en las distancias mayores, es probable que surjan problemas como el desvío (señales que llegan más tarde que el resto y debido a las demoras en las diferentes líneas.

No obstante, es regla general que para todos los métodos de transmisión de datos, la velocidad máxima posible sea inversamente proporcional a la distancia.

El segundo punto es lo que, de hecho, va por el cable (o cables) cuando se comunican los datos. Cuando enviamos el código binario "1" es transmitida una tensión que varía de -5 a -15, mientras que "0" es transmitido por una tensión que cambia de +5 a +15.

En el receptor, "1" es detectado entre -3 y -15, y "0" es detectado entre +3 y +15, y esto permite un margen de error si ocurre una pequeña caida de tensión.

Dijimos al comienzo que **RS232** era un estandard que no totalmente definido para el usuario final, y que potencialmente era más complejo que los otros dos. Ahora explicaremos la razón.

Los modos y medios por los cuales los elementos (en su sentido más

...nterfaces?

amplio) de un sistema se comunican, están gobernados por conjuntos de reglas, conocidas como protocolos.

Estos protocolos tienen un número de estratos funcionales, pero sólo tres de los siete se refieren al micro en solitario. Los restantes se refieren a sistemas de redes.

El estrato más bajo es el "estrato físico". Este define las características mecánicas (organización de las patillas, dimensiones de los enchufes y cableado), las características eléctricas (niveles de tensión, velocidades de transferencia, y algunos detalles sobre los circuitos), y las funciones de las señales (nombres y funciones).

En el siguiente nivel está el "estrato de enlace de datos", el cual define la manera en que los datos son transmitidos a través del estrato anterior. El problema es que el **RS232**, como estandard, se detiene repentinamente en el estrato físico, y en éste estrato existen muchas opciones.

A continuación describiremos los estratos:

Estrato físico: enchufes, patillas y cables. Estrato de enlace de datos: en serie, paridad y protocolos. Estrato de redes: conmutación y ronting. Estrato de transporte: comunica los estratos de arriba con los de abajo. Estrato de sesiones: comunica la aplicación al sistema. Estrado de presentación: conversión y formateado. Estrado de aplicación: programa de aplicación.

Una definición más, y pasaremos a describir la gama común de protocolos utilizados por el estrato de enlace de datos del **RS232**. Hasta ahora sólo hemos hablado del movimiento de los datos en una dirección "una interface unidireccional". Sin embargo, en muchos interfaces, los datos se mueven en ambas direcciones.

Si los datos pueden moverse en ambas direcciones, pero únicamente en un sentido en un instante particular, esto es "half-duplex". Si los datos pueden moverse en ambas direcciones simultáneamente, esto es "full-duplex".

IEEE 488

En 1975 se aprobó el estandard **IEEE 488** —el producto de cuatro años de trabajo en una estructura de interface, que podía enlazar no sólo periféricos sino sistemas completos. Una de las fuerzas principales impulsoras de este desarrollo era **Hewlett-Packard,** y ciertamente, el bus todavía es mencionado como **HPIB (Hewlett-Packard Interface Bus).**

Su función principal consistía en enlazar aparatos tan esotéricos como registradores cronológicos automáticos de datos, analizadores de espectro y voltímetros digitales. Para el usuario no técnico llegó cuando fue incluido en la gama de ordenadores de **Commodore.** En este caso, los dispositivos que comunicaban eran impresoras, discos, modems y otros periféricos más convencionales.

Sin embargo, la inclusión de este

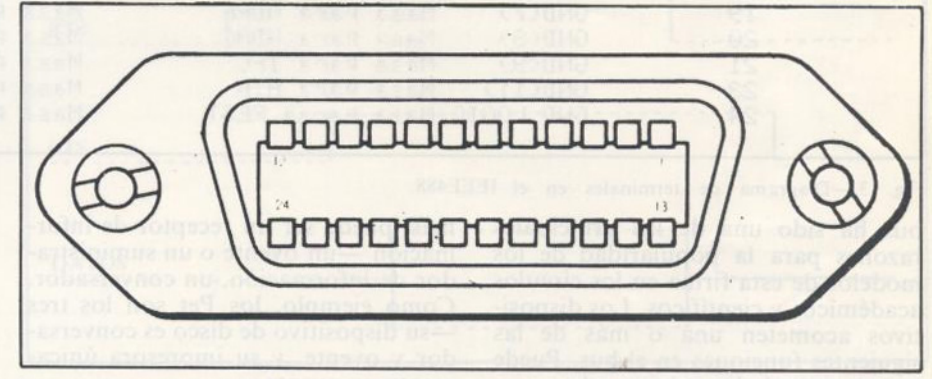


Fig. 1.—El conector GPIB.

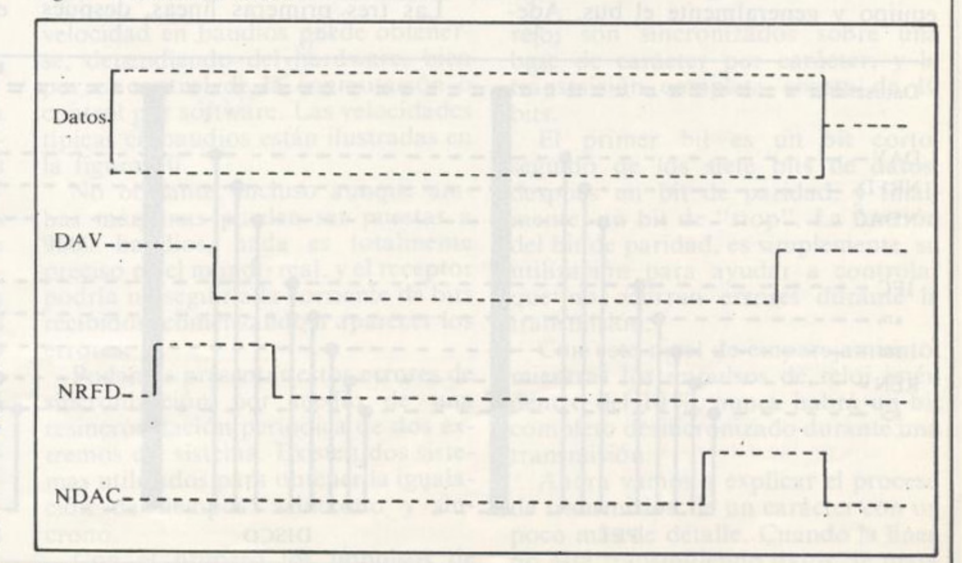


Fig. 2.—Control de los datos en el IEEE488 terminales.

PATILLA No.	CODIGO	SIGNIFICADO	FUNCION
1	DIO1	E/S de datos	Info de datos y direcciones
2	DIO2	E/S de datos	Info de datos y direcciones
3	DIO3	E/S de datos	Info de datos y direcciones
4	DIO4	E/S de datos	Info de datos y direcciones
5	EOI	End Or Identify	Fin de la transferencia
6	DAV	Data Valid	Datos estables
7	NRFD	Not Ready For Data	Indicador de listo
8	NDAC	Not Data Accepted	Falso si acepta datos
9	IFC	Interface Clear	"Reset" del interface
10	SRQ	Service Request	Dispositivo envia Peticion
11	ATN	Attention	Notificacion controlador
12	SHIELD	Cable apantallado	Masa del instrumento
13	DIO5	E/S de datos	Info de datos y direcciones
14	DIO6	E/S de datos	Info de datos y direcciones
15	DIO7	E/S de datos	Info de datos y direcciones
16	DIO8	E/S de datos	Info de datos y direcciones
17	REN	Remote Enable	Conmut. remoto/local
18	GND(6)	Masa Para DAV	Masa Para linea de control
19	GND(7)	Masa Para NDRF	Masa Para linea de control
20	GND(8)	Masa Para NDAC	Masa Para linea de control
21	GND(9)	Masa Para IFC	Masa Para linea de control
23	GND(11)	Masa Para ATN	Masa Para linea de control
24	GND LOGIC	Masa Panaa REST	Masa Para linea de control

Fig. 3.—Diagrama de terminales en el IEEE488.

bus ha sido una de las principales razones para la popularidad de los modelos de esta firma en los círculos académicos y científicos. Los dispositivos acometen una o más de las siguientes funciones en el bus. Puede ser un controlador gobernando a otro equipo y generalmente el bus. Además, puede ser un receptor de información —un oyente o un suministrador de información, un conversador. Como ejemplo, los Pet son los tres —su dispositivo de disco es conversador y oyente, y su impresora únicamente es oyente.

Las tres primeras líneas, después del bus de datos (ocho líneas si es una estructura en paralelo), controlan el "apretón de manos" de los datos. Las restantes gestionan el bus y sus funciones. El "apretón de manos" es acometido primero por un conversador, poniendo la pastilla DAV en estado alto, que indica la invalidez de

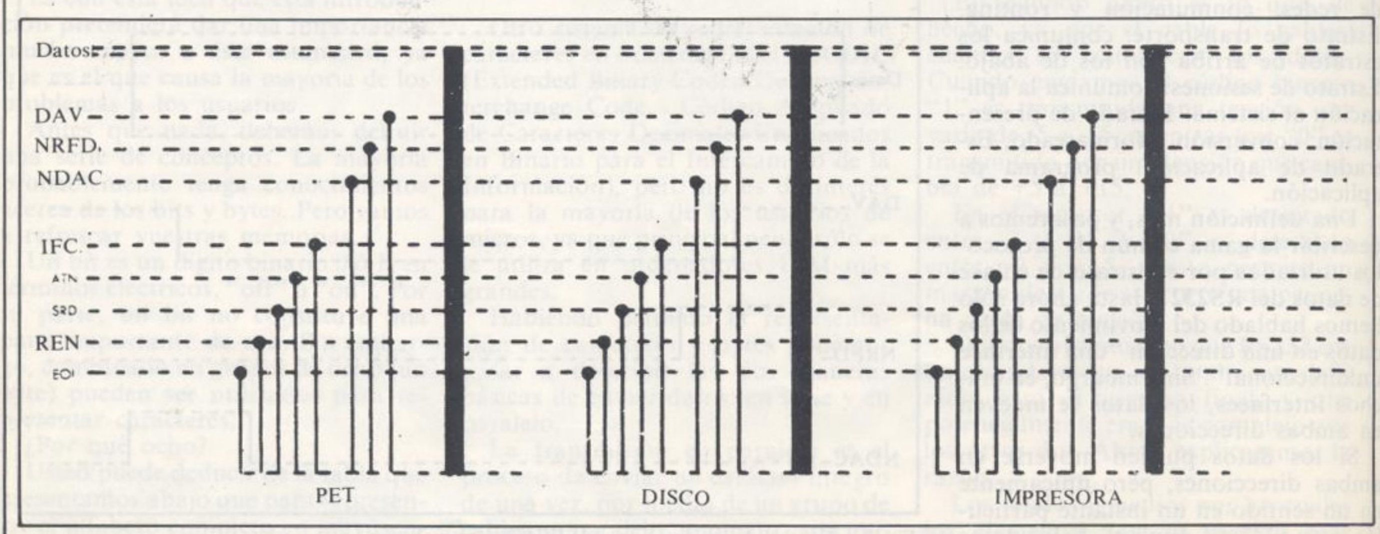


Fig. 4.—Líneas de control en el IEEE488.

los datos. Los datos se colocan en el bus. Cuando los oyentes han sido seleccionados los datos expresan su disposición liberando el NRFD (datos no disponibles). La pastilla DAV se desactiva y mantiene los datos estables hasta que todos los oyentes hayan recibido los datos, indicado por el NDAC yendo al estado bajo (datos no aceptados). En la figura 2 aparecen los impulsos que aparecen en función del tiempo.

Una de las ventajas más grandes del **IEEE 488** es la facilidad con que pueden establecerse las comunicaciones entre muchos dispositivos diferentes, pero utilizando siempre el mismo grupo de simples comandos del BASIC.

Como de costumbre, no todos los *interfaces* **IEEE 488** son totalmente estandard. Una versión no estandard es la que se utiliza en los Commodore. La diferencia fundamental estriba en los tiempos utilizados. Las señales tienen sincronizaciones ligeramente diferentes, y si una línea no realiza una conexión, el sistema tendrá un "compás de espera". En el **IEEE** standard de **Hewlett-Packard**, el *interface* sólo continuará siempre adelante. Estos cambios no afectarían demasiado a un **Commodore** a través de su **IEEE**.

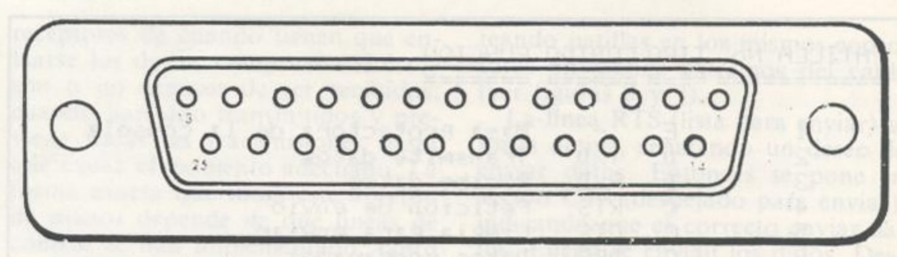


Fig. 5.—Designación de terminales en el RS232.



Fig. 6.—Transmisión de un caracter.

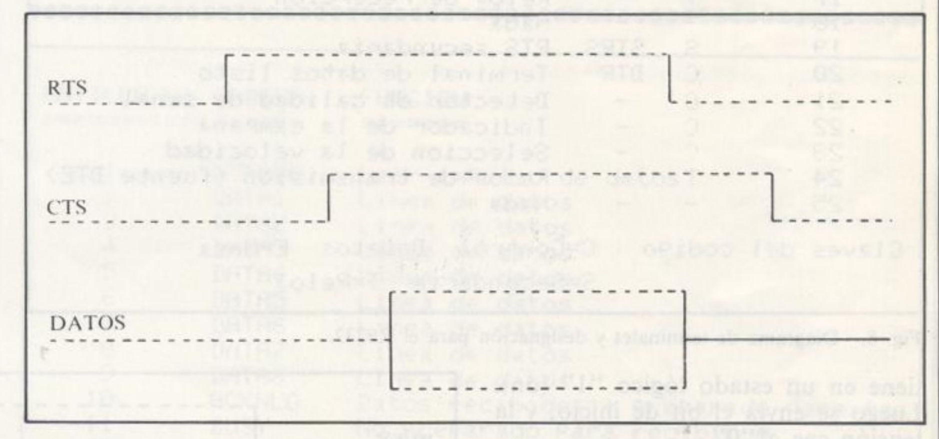


Fig. 7.—Apretón de manos con dos líneas de control.

RS 232

En las comunicaciones de datos la sincronización es de gran importancia. Si uno envía 3 bits "0", entonces la línea estaría alta durante un período de tiempo, y ese tiempo sería tres veces mayor que el que requiere un bit individual. Por consiguiente debemos conocer la velocidad de la transmisión, antes de que podamos determinar cuantos bits han sido enviados.

La temporización en el extremo receptor debe ser la misma que el de transmisión. La velocidad de transmisión es medida en baudios (normalmente equivalen a bits por segundo), de la cual existe una serie de ellas establecidas como estandares. Esta velocidad en baudios puede obtenerse, dependiendo del hardware, bien por el control de la conmutación o control por software. Las velocidades típicas en baudios están ilustradas en la figura 10.

No obstante, incluso aunque ambas máquinas pueden ser puestas a 9600 baudios, nada es totalmente preciso en el mundo real, y el receptor podría no seguir a la corriente de bits recibidos, comenzando a aparecer los errores.

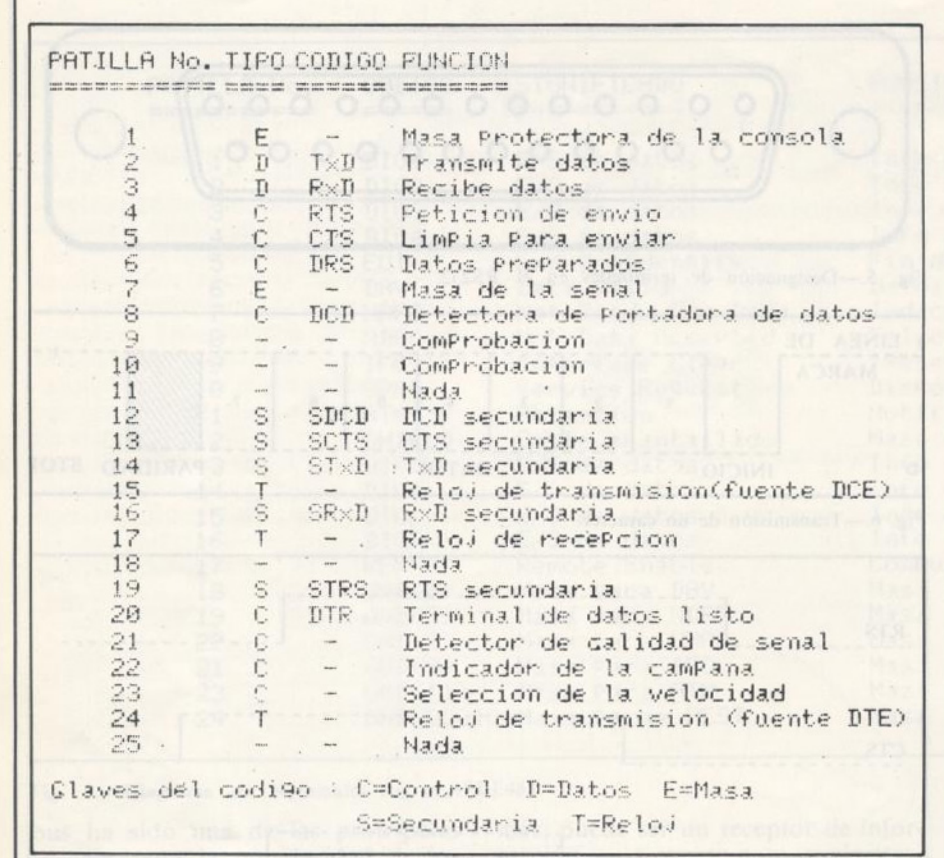
Podemos presentar estos errores de sincronización por medio de una resincronización periódica de dos extremos del sistema. Existen dos sistemas utilizados para obtener la igualación de tiempos: asíncrono y síncrono.

Con el primero los impulsos de reloj son sincronizados sobre una base de carácter por carácter, y la transmisión completa consta de 10 bits.

El primer bit es un bit corto, seguido de los siete bits de datos, después un bit de paridad, y finalmente, un bit de "stop". La función del bit de paridad, es simplemente, su utilización para ayudar a controlar que no ocurran errores durante la transmisión.

Con este nivel de emparejamiento, mientras los impulsos de reloj estén dentro del 10%, nunca habrá un bit completo desincronizado durante una transmisión.

Ahora vamos a explicar el proceso de transmisión de un carácter con un poco más de detalle. Cuando la línea no está transmitiendo datos, se man-



PATILLA No.	TIPO	CODIGO	FUNCION
1	E	-	Masa protectora de la consola
2	D	TxD	Transmite datos
3	D	RxD	Recibe datos
4	C	RTS	Peticion de envio
5	C	CTS	Limpia para enviar
6	C	DRS	Datos preparados
7	E	-	Masa de la senal
8	C	DCD	Detectora de portadora de datos
9	-	-	Comprobacion
10	-	-	Comprobacion
11	-	-	Nada
12	S	SDCD	DCD secundaria
13	S	SCTS	CTS secundaria
14	S	STxD	TxD secundaria
15	T	-	Reloj de transmision(fuente DCE)
16	S	SRxD	RxD secundaria
17	T	-	Reloj de recepcion
18	-	-	Nada
19	S	STRS	RTS secundaria
20	C	DTR	Terminal de datos listo
21	C	-	Detector de calidad de senal
22	C	-	Indicador de la campana
23	C	-	Seleccion de la velocidad
24	T	-	Reloj de transmision (fuente DTE)
25	-	-	Nada

Claves del codigo : C=Control D=Datos E=Masa S=Secundaria T=Reloj

Fig. 8.—Diagrama de terminales y designación para el RS232.

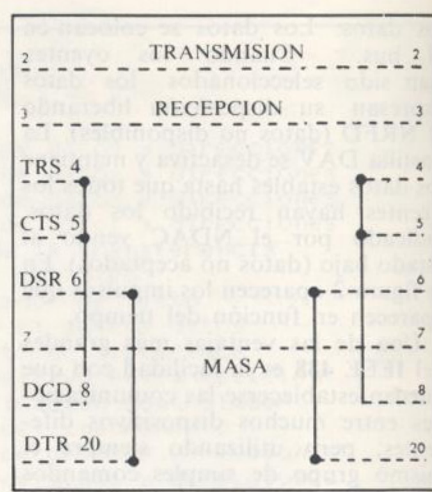
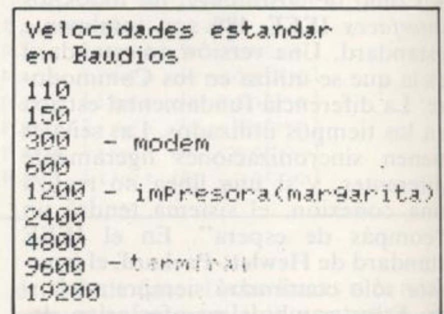


Fig. 9.—Apretón de manos.



Velocidades estandar en Baudios
110
150
300 - modem
600
1200 - impresora(margarita)
2400
4800
9600 - terminal
19200

Fig. 10.—Marcas estandarizadas en baudi.

tiene en un estado lógico "1" (on). Luego se envía el bit de inicio, y la tensión cae a "0".

Ahora podemos enviar los siete bits de datos, que pueden ser cualquier combinación de 0 a 1, correspondiente al carácter **ASCII** requerido. A continuación está el bit de paridad, y finalmente la línea es puesta en el estado lógico "1" por un bit de stop. (Ver figura 6). Este bit de stop, sin embargo, puede tener una longitud de 1,5 ó 2 bits, dependiendo del sistema.

Por otro lado, y en oposición a la sincronización carácter por carácter, la comunicación síncrona "envía el impulso" de reloj, junto con el dato.

En las distancias cortas se utiliza la técnica de un cable paralelo, que transporte una señal de reloj separada, pero en las distancias largas y vía modems, debe utilizarse la técnica incluirla.

En este caso la señal de reloj es codificada con los datos a través de

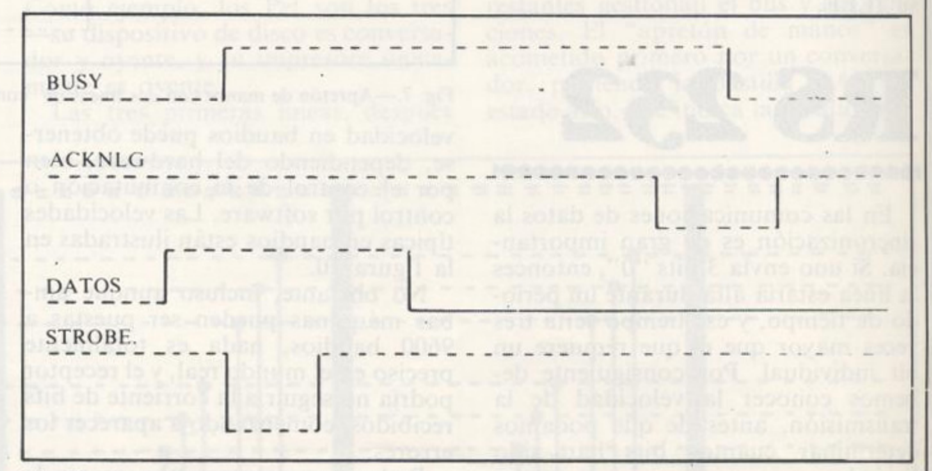


Fig. 12.—Apretón de manos en el Centronics.

un modems síncrono, de manera similar a la que utilizan las emisoras de radio para emitir en estéreo.

En el extremo receptor, la señal es decodificada, apareciendo como si el sistema de cable paralelo estuviera en funcionamiento. Además, con la comunicación síncrona son identificados los bloques completos de datos y no sólo caracteres.

Por consiguiente, debido a este modo de bloques, en lugar de en

caracteres, se reduce el número de bits redundantes.

Sólo siete de los 10 bits, enviados asíncronamente, son datos. Esto es, el 30 % de los bits no son "de información". No obstante (en el modo síncrono), especialmente con mensajes largos, menos del 3 % de los bits no son de información.

La paridad es un simple sistema de detección de errores. Como hemos dicho antes, los datos son enviados como series de bits, por lo tanto la "A" mayúscula sería "1000001" (ver figura 11).

Se agrega un bit de paridad (dependiendo de como se establezca el control), para hacer que el número total de "0"s y "1"s sea impar o par. El control de paridad sólo señalará los errores de uno o un número par o impar de cambios en bits. Un número par de errores preservará el cambio original de paridad.

Ahora, para dejar la comprobación de errores y comenzar con el apretón de manos en profundidad, nos referiremos al *interface* para una impresora de margarita (XON/XOFF y ETX/ACK). Estas impresoras, que manejan unos cincuenta caracteres por segundo, tienen dos velocidades en baudios, 300 y 1200. Supongamos que no deseamos ejecutar a 300, ya que esto disminuiría la velocidad del dispositivo considerablemente.

El *port* de la impresora recibe 120 cps (caracteres por segundo), mientras que sólo imprime a 50 cps. Se produce un cuello de botella.

En XON/XOFF cuando el número de caracteres sobrepasa un cierto límite superior, la transmisión se detiene y únicamente se reanuda cuando el número disminuye hasta otro límite inferior. ETX/ACK es similar, pero el carácter de transmisión preciso de hecho es enviado como el último carácter en cada bloque transmitido.

Una mirada a las señales de las patillas y la tabla de funciones del **RS232**, especialmente aquellas marcadas con "C", proporcionará al lector una idea acerca de lo que es el apretón de manos. Es la técnica por la cual el paso de datos puede ser gestionado y controlado eficientemente.

Este apretón de manos alerta a los receptores de cuando tienen que enviarse los datos, comprueba si éstos son o no capaces de ser recibidos, cuando han sido transmitidos y previene todas las transmisiones hasta que exista el momento adecuado. La forma exacta que toma este apretón de manos depende de qué líneas de control se han implementado, tanto en el diseño del *hardware* como en el de los cables, de lo cual existen muchas versiones.

Un número de cables implementan su propio apretón de manos puenteando patillas en los mismos conectores en ambos extremos del cable (ver figuras 9 y 7).

La línea RTS (lista para enviar) se torna activa, señalando un deseo de enviar datos. Entonces se pone en acción CTS (despejado para enviar), indicando que es correcto enviar datos. Luego se envían los datos. Después RTS es desactivado, informando al sistema que no desea enviar ningún dato más. La transmisión de los datos se completa y CTS retorna a su estado inactivo.

CENTRONICS

PATILLA No.	CODIGO	FUNCION
1	STROBE	Leer impulso de datos
2	DATA1	Linea de datos
3	DATA2	Linea de datos
4	DATA3	Linea de datos
5	DATA4	Linea de datos
6	DATA5	Linea de datos
7	DATA6	Linea de datos
8	DATA7	Linea de datos
9	DATA8	Linea de datos
10	ACKNLG	Datos recibidos + Preparado Para mas
11	BUSY	No Preparado Para recibir *
12	PE	Falta Papel
13	+5v	
14	AUTO FEED	Indicador de linea extra
15	NC	No conectada
16	GND LOGIC	Masa Para la logica
17	GND CASE	Masa del chasis
18	NC	No conectada
19-30	GND	Senal de masa (cable bifilar)
31	INT	Puesta a cero y limpieza del buffer
32	ERROR	Ver BUSY
33	GND	Masa Para la senal
34	NC	No conectada
35	+ 5v	
36	SLCT IN	Codigo DC1/DC3 oPcional

*El BUSY es activado si:
1. Se estan recibiendo datos.
2. Si la impresora esta imprimiendo
3. La impresora esta "off-line"
4. Si esta Presente una condidion de error

Fig. 14.—Interface Centronics.

Como dijimos anteriormente, los *interfaces* tales como el **Centronics** no ofrecen las mismas posibilidades de error que el **RS232.** Por lo tanto, nos referiremos únicamente al apretón de manos y la descripción de patillas y señales.

Este produce el BUSY (ocupado), manteniéndose en estado alto, mientras se estén aceptando datos. La línea ACKNLG (reconocimiento) se mantiene en estado bajo, indicando que los datos han sido aceptados. Como contrapartida, pone la línea BUSY en estado bajo, tal como se aprecia en la figura 12.

Fig. 13.—Conector y numeración de los terminales.

go, no hay que pasar por alto lo que está en el medio: el cable.

Calcule qué señales están siendo utilizadas y asegúrese de que dispone de la conexión apropiada. Mucha gente gasta más de lo que necesita en cables. El coste de un cable está directamente relacionado a cuentos enlaces se necesitan, por lo tanto una conexión completa de 25 líneas es mucho más costosa que de 3. Por lo tanto, compruebe lo que necesita y pídalo —no hay motivo para pagar cables por los cuales nunca pasarán señales.

Los poseedores de Pet, de Commodore y otro *hardware* que soporta el *interface* **IEEE 488,** deberían, si es que aún no lo han hecho, observar con mayor atención algunas de las funciones que ofrece.

El científico debe tener en cuenta aún más la facilidad con la cual la máquina puede ser enlazada con una amplia variedad de sistemas, tanto de captura de datos como de informes con los cuales tiene que lidiar. Un sistema que utiliza un *interface* de tal clase puede conectarse, realizar estadísticas, informes en un plotter y operar el sistema de control. Pudiendo programarse toda esta gestión en BASIC, sin necesidad de volver al lenguaje máquina (a no ser que se requiera una respuesta muy rápida).

Los propietarios de ordenadores con *interface* **Centronics** y una impresora compatible (la mayoría de las matriciales, sean baratas o caras) siempre deberían utilizar esta opción y estar agradecidas de que dispongan de una solución sencilla.

CONCLUSION

Estos, entonces, son los tres *interfaces* que encontrará el usuario de ordenadores pequeños. Podrá encontrar el **RS232** en todos o casi todos los microordenadores. Es capaz de enviar y recibir información con una amplia variedad de periféricos, como impresoras, terminales, lectoras de códigos de barras y modems. En muchos de estos periféricos **RS232** es, ciertamente, el único *interface* disponible. Resumiendo, entre las variables implicadas en la conexión de un periférico al **RS232,** aquéllas a tener en cuenta son:

- Velocidad de baudios — ¿Están ambos extremos transmitiendo a la misma velocidad?
- Paridad — ¿Está activado el control de paridad (sino ir a 4)?
- Tipo de paridad — ¿Es la paridad par o impar?
- Protocolo — ¿Es DTR. XON/XOFF, etc.?

Esta lista debería cubrir a la mayoría de los sistemas, pero algunos introducen aún más variables, dependiendo del *hardware.*

- Bits de Stop — ¿Es 1, 1,5 ó 2?
- Longitud del Byte — ¿7 ú 8 bits?

Esos seis factores, como lista de comprobación, tendrían que cubrir los requerimientos disponibles en el ordenador y el periférico. Sin embar-

Fig. 11.—Ilustración de la comprobación de paridad.

ILUSTRACION DEL CONTROL DE PARIDAD

CARACTER	ASCII	EN BITS	BIT DE PARIDAD IMPAR	No. "1"s	PAR	No. "1"s
A	65	1000001	1	3	0	2
B	66	1000010	1	3	0	2
C	67	1000011	0	3	1	4
X	88	1011000	0	3	1	4
Y	89	1011001	1	5	0	4
Z	90	1011010	1	5	0	4

SU PROGRAMA PARA CUALQUIER SISTEMA COMMODORE PUEDE HACERLE GANAR 5.000 PTAS.

EL PRESENTE CONCURSO ESTA ABIERTO A TODOS NUESTROS LECTORES Y SU PARTICIPACION E INSCRIPCION ES GRATUITA. LEA LAS BASES DEL CONCURSO

- NO SE ESTABLECEN LIMITACIONES EN CUANTO A EXTENSION, TEMA ELEGIDO O MODELO DE ORDENADOR
- LOS CONCURSANTES DEBERAN ENVIARNOS A LA DIRECCION QUE FIGURA AL PIE, EL CASSETTE O DISKETTE CONTENIENDO EL PROGRAMA, UNA EXPLICACION DEL MISMO Y, AL SER POSIBLE, UN LISTADO EN PAPEL DE IMPRESORA. SE PODRAN ENVIAR TANTOS PROGRAMAS COMO SE DESEE
- LOS PROGRAMAS, PREVIA SELECCION, SERAN PUBLICADOS EN LA REVISTA, OBTENIENDO TODOS ELLOS 5.000 PTAS.
- LA DECISION SOBRE LA PUBLICACION O NO DE UN PROGRAMA CORRESPONDE UNICAMENTE AL JURADO NOMBRADO AL EFECTO POR "COMMODORE MAGAZINE", SIENDO SU FALLO INAPELABLE
- LOS CRITERIOS DE SELECCION SE BASARAN EN LA CREATIVIDAD DEL TEMA ELEGIDO Y LA ORIGINALIDAD Y/O SENCILLEZ EN EL METODO DE PROGRAMACION GLOBAL
- ENVIAR A: CONCURSO COMMODORE MAGAZINE



commodore Magazine

C/Bravo Murillo, 377. Madrid -20



64 musical

Con este programa podrás transformar el teclado de tu **Commodore 64** en un órgano electrónico con nueve instrumentos. El tono de los distintos instrumentos se consigue "Pokeando" W (dirección 54276) con el valor adecuado de la forma de onda correspondiente. Hay que decir que hace falta un poco de imaginación para reconocer algunos instrumentos, pero está en tus manos el cambiarlos por los que quieras.

Además el **"64 MUSICAL"** te servirá para conocer algunas de las posibilidades, en cuanto a sonido se refiere, de tu ordenador, sin tener que recurrir a los complicados POKEs.

Para que puedas efectuar las modificaciones que desees con objeto de "oir" de lo que es capaz tu **C64**, aquí tienes algunas variables de interés utilizadas por el programa:

A: Ataque/decaimiento.
V: Volumen.
S: Sostenimiento/relajación.
W: Forma de onda.
H y L: Frecuencias alta y baja del filtro.

```
0 GOSUB200:POKE53280,0:POKE53281,12:PRINT"⌐"
1 POKEV,15:POKEL,0:POKEH,0:PRINT"⌐▒▒────────────────────────────────"
2 PRINT"▒▒ ▌▌ ▌▌ | ▌▌ ▌▌ ▌▌ | ▌▌ ▌▌ | ▌▌ ▌▌ "
3 PRINT"▒ ▌▌ ▌▌ | ▌▌ ▌▌ ▌▌ | ▌▌ ▌▌ | ▌▌ ▌▌ "
4 PRINT"▒ ▌▌ ▌▌ | ▌▌ ▌▌ ▌▌ | ▌▌ ▌▌ | ▌▌ ▌▌ "
5 PRINT"▒ | | | | | | | | | | | | | "
6 PRINT"▒Q|W|E|R|T|Y|U|I|O|P|@|*|↑"
7 PRINT"■▒────────────────────────────"
8 PRINT"▒▒C|D|E|F|G|A|B|C|D|E|F|G|A"
9 PRINT"▌▒▒▒P■ ▒PIANO▌▌▌▌▌▌▌▌▌▌▌▒▒C■ ▒CLAVICORDIO":PRINT"▒▒▒O■ ▒ORGANO"
10 PRINT"⌐▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▒▒U■ ▒TUBA":PRINT"▒▒▒▒F■ ▒FLAUTA▌▌▌▌▌▌▌▌▌▒▒A■ ▒ACOR";
11 PRINT"DEON":PRINT"▒▒▒B■ ▒OBOE▌▌▌▌▌▌▌▌▌▌▌▒▒T■ ▒TROMPETA"
12 PRINT"▒▒▒▒X■ ▒XILOFONO"
13 PRINT"▒▒▒CUAL QUIERES? ▒"
15 GETB$:IFB$="" THEN15
16 GETA$
18 IFB$="P"THENGOSUB100
20 IFB$="F"THENGOSUB106
22 IFB$="B"THENGOSUB112
24 IFB$="C"THENGOSUB118
26 IFB$="O"THENGOSUB124
28 IFB$="U"THENGOSUB130
30 IFB$="T"THENGOSUB136
32 IFB$="A"THENGOSUB142
34 IFB$="X"THENGOSUB148
40 IFA$="Q"THENPOKEH,34:POKEL,75:GOSUB90
42 IFA$="2"THENPOKEH,36:POKEL,85:GOSUB90
44 IFA$="W"THENPOKEH,38:POKEL,126:GOSUB90
46 IFA$="3"THENPOKEH,40:POKEL,200:GOSUB90
48 IFA$="E"THENPOKEH,43:POKEL,52:GOSUB90
50 IFA$="R"THENPOKEH,45:POKEL,198:GOSUB90
52 IFA$="5"THENPOKEH,48:POKEL,127:GOSUB90
54 IFA$="T"THENPOKEH,51:POKEL,97:GOSUB90
56 IFA$="6"THENPOKEH,54:POKEL,111:GOSUB90
58 IFA$="Y"THENPOKEH,57:POKEL,172:GOSUB90
60 IFA$="7"THENPOKEH,61:POKEL,126:GOSUB90
62 IFA$="U"THENPOKEH,64:POKEL,188:GOSUB90
64 IFA$="I"THENPOKEH,68:POKEL,149:GOSUB90
```

```
66 IFA$="9"THENPOKEH,72:POKEL,169:GOSUB90
68 IFA$="O"THENPOKEH,76:POKEL,252:GOSUB90
70 IFA$="0"THENPOKEH,81:POKEL,161:GOSUB90
72 IFA$="P"THENPOKEH,86:POKEL,105:GOSUB90
74 IFA$="@"THENPOKEH,91:POKEL,140:GOSUB90
76 IFA$="-"THENPOKEH,96:POKEL,254:GOSUB90
78 IFA$="*"THENPOKEH,102:POKEL,194:GOSUB90
80 IFA$="£"THENPOKEH,108:POKEL,223:GOSUB90
82 IFA$="↑"THENPOKEH,115:POKEL,88:GOSUB90
84 IFA$="N"THENGOTO1
85 IFA$="F"THENPOKEH,0:POKEL,0:END
86 GOTO16
90 POKEA,0:POKES,0:POKEW,0:POKEL,0
94 RETURN
100 POKEA,9:POKES,0:POKEW,65
103 POKE54275,0:POKE54274,255
105 RETURN
106 POKEA,64:POKES,128
107 POKEW,17
110 RETURN
112 POKEA,190:POKES,0
114 POKEW,17
116 RETURN
118 POKEA,9:POKES,0
120 POKEW,33
122 RETURN
124 POKEA,0:POKES,240
126 POKEW,33
128 RETURN
130 POKEA,0:POKES,240
132 POKEW,17
134 RETURN
136 POKEA,96:POKES,1
137 POKEW,33
140 RETURN
142 POKEA,102:POKES,0
144 POKEW,17
146 RETURN
148 POKEA,9:POKES,0
150 POKEW,17
152 RETURN
200 V=54296:A=54277:S=54278:W=54276:H=54273:L=54272
201 POKE53280,0:POKE53281,0:POKEL,0:POKEH,0:POKEW,0
202 PRINT"▯▯▯▯▯▯▯▯▯▯ ▮▮ ▮▮ | ▮▮ ▮▮ ▮▮ | ▮▮ ▮▮ | ▮▮ ▮▮ ▮▮ "
203 PRINT"▯▯▯▯▯▯▯▯ ▮▮ ▮▮ | ▮▮ ▮▮ ▮▮ | ▮▮ ▮▮ | ▮▮ ▮▮ ▮▮ "
204 PRINT"▯▯▯▯▯▯▯▯ ▮▮ ▮▮ | ▮▮ ▮▮ ▮▮ | ▮▮ ▮▮ | ▮▮ ▮▮ ▮▮ "
206 PRINT"▯▯▯▯▯▯▯▯ | | | | | | | | | | | | | | "
208 CO=54272:PRINT"▯▯▯▯▯▯▯▯C|O|M|M|O|D|O|R|E| | |6|4| "
209 FORJ=0TO12:FORI=55502TO55528:POKEI+J*40,12:POKEI+J*40-CO,160:NEXTI,J:PRINT"▯
210 PRINT"▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▒   ▒    "
212 PRINT"▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▒   ▒    "
214 PRINT"▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▮▮▮▮▮ ▮ ▮ "
216 PRINT"▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ ▮ ▮▮▮▮▮▮▮"
218 PRINT"▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▮▮▮▮▮    ▮ "
220 PRINT"▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯*MU*SI▮CA▯L*▮
222 PRINT"▯"
224 PRINT"▯▯▯▯▯▯▯▯C|O|M|M|O|D|O|R|E| | |6|4| "
226 PRINT"▯▯▯▯▯▯▯▯ | | | | | | | | | | | | | | "
228 PRINT"▯▯▯▯▯▯▯▯ ▮▮ ▮▮ | ▮▮ ▮▮ ▮▮ | ▮▮ ▮▮ | ▮▮ ▮▮ ▮▮ "
```

Programas

Viene de la página anterior

```
230 PRINT"   | | |   | |   | | |"
232 PRINT"   | | |   | |   | | |"
234 PRINT"PULSA CUALQUIER TECLA"
235 GETA$:IFA$=""THENGOTO235
236 PRINT"":POKE53280,0:POKE53281,0
237 PRINTTAB(15)"64 MUSICAL"
239 PRINT"EL 64 MUSICAL TIENE 9 INSTRUMENTOS."
240 PRINT"LAS NOTAS EMPIEZAN DESDE LA C-MEDIA."
243 PRINT"LAS TECLAS A USAR SON:"
244 PRINT"Q=C      2=C#      W=D      3=D#
246 PRINT"E=E      R=F       5=F#
248 PRINT"T=G      6=G#      Y=A      7=A#
250 PRINT"U=B      I=C       9=C#     O=D
252 PRINT"0=D#     P=E       @=F      -=F#
254 PRINT"*=G      £=G#      ↑=A"
256 PRINT"F=FIN                 N=NUEVO INSTRUMENTO"
260 PRINT"PULSA CUALQUIER TECLA PARA EMPEZAR"
262 GETA$:IFA$=""THENGOTO262
264 RETURN
```



Andromeda

Naves exploradoras han detectado el avance de una flota de combate desde el vacio exterior a nuestra galaxia, con origen según el vector dirección de la galaxia a Andromeda. Para todas las naves de patrulla, las órdenes son localizar y destruir cualquier nave que, enviada en avanzadilla, traspase los límites de seguridad del sector. Las instrucciones de combate van incluidas en el programa que se adjunta, que deberá ser introducido inmediatamente en el ordenador **VIC 20** de abordo, después de haber conectado el módulo de expansión de memoria de 3K. En cualquier encuentro el factor tiempo es esencial, por lo que será preciso actuar con la mayor rapidez, para no traspasar el límite de tiempo. Además, según informes de las naves exploradoras, el color de las naves enemigas puede variar según el tipo de campo de fuerza que emplean como protección, por lo que su destrucción será más completa en unos casos que en otros, según el color que presenten al ser alcanzadas.

El ordenador de abordo llevará la cuenta de los puntos conseguidos.

VIC 20 + EXPANSION DE 3K

```
0 CLR:RESTORE:GOSUB10000
1 MZ=13:MX=1:REM**ANDROMEDA**
2 DIM E(1,MZ),D(4)
5 GOSUB1000:FORI=1TO4:READD(I):NEXT
7 PRINT"□■";
8 GOSUB300
9 POKE36879,8:POKE650,255:POKE36878,15:TI$="000000":GOSUB400
10 FOR I=1 TO MX
11 IF MX=13THEN 1100
12 PRINT"■PT:"SC"■■■■■TI:"INT(TI/60)
15 IF VAL(TI$)>30+(MX*7) THEN 1200
17 IF NG=1 THEN NG=0:GOSUB 900: GOSUB 1000:GOTO9
20 IF E(0,I)=0 THEN GOSUB100
30 GETA$
35 IF PEEK(198)>2 THEN POKE198,2
40 IF A$<>"" THEN GOSUB 200
88 M=0
89 GOSUB 300
90 NEXTI
99 GOTO 10
100 REM MUEVE MARCIANOS
105 GOSUB 600
110 ON INT(RND(1)*4)+1 GOTO 120,125,130,135
120 E(1,I)=E(1,I)+22:GOTO140
125 E(1,I)=E(1,I)-22:GOTO150
130 E(1,I)=E(1,I)-1:GOTO150
135 E(1,I)=E(1,I)+1:GOTO140
140 IF E(1,I)<7724 THEN E(1,I)=E(1,I)+22
150 IF E(1,I)>8185 THEN E(1,I)=E(1,I)-22
155 POKE 36874,128+I*10
160 GOSUB 700
180 POKE 36874,0
199 RETURN
200 REM COMANDO
210 IF A$=CHR$(13) THEN GOSUB 500
220 IF A$="Y" THEN M=-22:GOTO280
230 IF A$="B" THEN M=+22:GOTO280
240 IF A$="G" THEN M=-1:GOTO 280
250 IF A$="H" THEN M=+1:GOTO280
260 GOTO 299
280 GOSUB 300
299 RETURN
300 REM MOVIMIENTO DE TRAVES
310 POKEY,32
320 Y=Y+M
330 POKEY,86
399 RETURN
400 REM COLORES
402 PRINT"□";:FORI=1 TO 10
403 PRINT"■";SPC(I);"■ ESPERA..."
404 FOR P=0 TO 50: NEXT: NEXT:PRINT"□";
405 POKE 36876,0
410 FOR I=1 TO 4
420 FOR J=0 TO 110
430 POKE 38400+110*I+J,I+1
440 NEXT:NEXT
499 RETURN
500 REM DISPAROS
```



Viene de la página anterior

```
510 F1=Y-126:F2=Y-138
520 F3=Y+126:F4=Y+138
530 FORK=1 TO 6
540 POKE F1,32:POKEF2,32:POKEF3,32:POKEF4,32
550 POKE 36876,200-K*4
554 REM
555 REM
558 REM
559 F1=F1+21:F2=F2+23:F3=F3-21:F4=F4-23
560 POKE F1,78:POKEF2,77:POKEF3,78:POKEF4,77
570 NEXT:POKE36876,0
580 FORJ=1 TO MX
582 IF E(0,J)<>0 THEN 586
584 IF E(1,J)=Y THEN GOSUB 800
586 NEXT:POKE 36876,128+K*5: IF NG=1 THEN SC=SC+100
590 POKE F1,32:POKEF2,32:POKEF3,32:POKEF4,32
595 POKE 36876,0
599 RETURN
600 REM*BORRA TODO*
610 POKEE(1,I),32
620 POKEE(1,I)-1,32
630 POKE E(1,I)+1,32
699 RETURN
700 REM*DIBUJA INVASORES*
710 POKEE(1,I),81
720 POKEE(1,I)-1,60
730 POKEE(1,I)+1,62
799 RETURN
800 REM*NAVE ALCANZADA*
810 E(0,J)=1
820 POKE36877,200:FORJK=1TO30
830 POKEE(1,J),42:POKEE(1,J)-1,42:POKEE(1,J)+1,42
832 NEXT:POKE36877,0
835 POKEE(1,J),32:POKEE(1,J)-1,32:POKEE(1,J)+1,32
840 SC=SC+PEEK(Y+30720)*2
850 REM
860 FORW=1TOMX
870 IF E(0,W)=1 THEN BN=BN+1
880 NEXT
890 IF BN>=MX THEN NG=1
895 BN=0
899 RETURN
900 REM*OTRO JUEGO*
910 FORP=1 TO1000:NEXT
915 IF MX=12 THEN GOSUB1300:GOSUB1400:GOTO1100
920 PRINT"]QQQQQLO CONSEGUISTE"
930 PRINT"QPERO AHORA VA A SER"
940 PRINT"QMAS DIFICIL!"
950 MX=MX+1
960 PRINT"IQAHORA SON"MX"INVASORES"
970 FORP=1 TO 5000:NEXT
980 TI$="000000"
999 RETURN
```

```
1000 REM*INICIO*
1010 FOR I=1 TO  MX
1020 E(1,I)=7680+INT(RND(1)*400)+22:E(0,I)=0
1030 NEXT
1050 Y=7910
1055 PRINT""
1099 RETURN
1100 PRINT"LIMPIASTE"
1110 PRINT"LA GALAXIA"
1115 PRINT"DE INVASORES"
1120 PRINT"POR ELLO"
1130 PRINT"TUS PUNTOS SE"
1137 PRINT"MULTIPLICAN POR"
1138 PRINT"12"
1140 PRINT"LO QUE TE DA"SC*12:SC=SC*12
1150 PRINT"GRACIAS POR JUGAR"
1160 GOTO 1299
1200 PRINT"FINFINFINFINFINFINFIN"
1210 PRINT"FINFINFINFINFINFINFIN"   :POKE8185,33
1212 FORI=1TO15
1215 POKEY-22,42:POKEY+22,42:POKEY-1,42:POKEY+1,42
1220 FORL=1TO200:NEXTL
1225 POKEY-22,32:POKEY+22,32:POKEY-1,32:POKEY+1,32
1227 POKEY-23,42:POKEY+21,42:POKEY-21,42:POKEY+23,42
1230 FORL=1TO200:NEXTL
1240 POKEY-23,32:POKEY+21,32:POKEY-21,32:POKEY+23,32
1245 POKE36878,15-I:POKE36877,160-I
1250 NEXTI
1260 POKE36878,0:POKE36877,0
1270 PRINT"LIMPIASTE"
1271 PRINT""MX-1"DOCENAS"
1275 PRINT"DE INVASORES"
1280 PRINT"ASI QUE TU PUNTUACION"
1285 PRINT"ES";:SC=SC*MX:PRINTSC;"."
1290 PRINT"GRACIAS POR JUGAR"
1299 POKE198,0:FORI=1 TO 8000:NEXT:PRINT"";:POKE36879,27:END
1310 POKE 36876,0:RETURN
1400 GOSUB1500:D=1:Y=7703:FORI=160 TO 130 STEP-.05
1410 POKE 36877,I
1420 POKEY,160:Y=Y+D(D):POKEY,86:IF PEEK(Y+D(D))=160 THEN D=D+1:IF D>=5 THEN D=1
1430 NEXT:POKE36877,0:DATA1,22,-1,-22
1499 RETURN
1500 FORI=0TO21
1510 POKE7680+I,160
1520 POKE 7701+I*22,160
1530 POKE 8185-I,160
1540 POKE 8164-I*22,160
1550 NEXT
1560 RETURN
10000 PRINT"LAS TECLAS:"
10001 PRINT"Y-ARRIBA"
10002 PRINT"B-ABAJO"
10003 PRINT"G-IZQUIERDA"
10004 PRINT"H-DERECHA"
10005 PRINT"RETURN-FUEGO"
10006 PRINT"PUNTOS SEGUN COLOR"
10007 PRINT"TIEMPO LIMITE-130 SG."
10008 PRINT"PULSA UNA TECLA":WAIT203,191:GETA$:RETURN
```

Cómo diseñar juegos para ordenador (capitulo 2)



Concepción básica de los juegos inteligentes

Los juegos inteligentes, es decir, aquellos en que el ordenador intenta imitar la habilidad mental de una persona, basan su complejidad en el proceso de pensar y no en otros como puede ser la velocidad de respuesta, que se considera un parámetro secundario. Este problema radica en que no sólo necesitamos explicarle a la máquina cómo se juega, sino que además se le debe explicar qué hay que hacer para ganar (¿alguien está interesado en un programa que pierda siempre?), siendo este el principal problema a la hora de realizar un programa de este tipo.

Vamos a explicar esta idea con un ejemplo sencillo y que todas aquellas personas que vieron la películas "JUEGOS DE GUERRA" recordarán. Se trata del tres en raya americano. En este juego hay que colocar tres X (o 0) en fila en columna o en diagonal, la peculiaridad consiste en que una vez puesta una pieza no se puede quitar y se queda fija para toda la partida (que, por cierto, dura muy poco) y se siguen poniendo más por ambos bandos (una vez cada uno) hasta que uno gana o se llena el tablero. Un programa básico que juegue debe generar la casilla donde va a mover del modo más sencillo posible (al azar), evitar poner en una casilla ocupada y ver si alguno de los dos jugadores ha ganado o el tablero está lleno. El organigrama correspondiente es el de la figura 6. El programa ya realizado es el de la figura 7. Si introduce este programa y juega, verá que su juego es bastante irregular y no es difícil ganarle, por lo que perderá interés rápidamente, por lo tanto hay que hacer que la máquina piense más.

Algoritmos de juego

Para aquellas personas que no se consideran expertas en matemáticas hay que explicar que un algoritmo es una fórmula o proceso matemático aplicable para obtener solución a un problema específico. Este proceso tiene que estar explicado sin ambigüedades. El programa anterior hemos visto que tiene varios problemas: no comprueba la validez de las jugadas del contrario y se le gana fácilmente. El primero punto es fácil de resolver y sólo hace falta introducir una comprobación de que la casilla elegida es correcta. Es el segundo punto el que plantea los mayores problemas, porque los verdaderos juegos interesantes no tienen una solución precisa y definida, los que sí la tienen (por ejemplo el NIM) dejan de ser interesantès cuando el jugador la aprende, de modo que nuestra labor parece imposible al no existir solución. Esto no es totalmente correcto, no existe solución pero puede haber algoritmos que se acerquen a ella, de este modo el ordenador juega mejor pero puede perder, que es la situación real.

En nuestro caso mejoraremos el programa introduciendo un algoritmo que: si en el siguiente turno el jugador humano hace tres en raya, lo evite y que calcule cuál es su casilla más favorable. Es en este segundo punto donde el algoritmo deja de ser preciso, ¿cómo se sabe si una casilla es más favorable que las demás? Esto es algo que siempre queda a decisión del jugador, pero daremos unas ideas. Hay que tener en cuenta la utilidad para ganar de colocar una ficha en

esa posición; si colocando una ficha en esa casilla resulta que hacemos tres en raya, pues... de cabeza a ella. Si el caso anterior no se da podremos elegir la situación en que al colocarla evitamos que el contrario haga tres en raya, si no existe este peligro elegiremos la casilla en que obtengamos dos en raya y quede una tercera casilla en fila vacía (fíjese que pueden ser dos casillas nuestras separadas por una en blanco) y si no existe tampoco esta

Figura 7

```
5 REM ** INICIO **
10 PRINT "J"
15 REM ** LEER PRIMER JUGADOR **
20 INPUT "(1=MAQUINA, 2=HUMANO)";A
40 IF (A<>1) AND (A<>2) THEN GOTO 10
45 REM ** ELEGIR TURNO **
50 IF A=2 THEN GOSUB 500
60 IF A=1 THEN GOSUB 1000
65 REM ** COLOCAR FICHA **
70 T(F,C)=A
75 REM ** IMPRIMIR TABLERO **
77 PRINT "   123"
80 FOR I=1 TO 3
85 PRINT I;
90 FOR J=1 TO 3
100 IF T(I,J)=1 THEN PRINT "O";: GOTO 130
110 IF T(I,J)=2 THEN PRINT "X";:GOTO 130
120 PRINT " ";
130 NEXT J
140 PRINT
150 NEXT I
160 GOSUB 300
170 IF R3<>0 THEN GOSUB 250
180 REM ** TABLERO LLENO? **
190 FOR I=1 TO 3: FOR J=1 TO 3
200 IF T(I,J)=0 GOTO 230
210 NEXT J:NEXT I
220 REM ** FICHERO LLENO **
225 PRINT "TABLAS ***********":GOTO 10000
230 REM ** CAMBIAR TURNO **
240 IF A=1 THEN A=2: GOTO 45
245 A=1 :GOTO 45
250 REM ** TRES EN RAYA **
260 IF A=1 THEN PRINT ">>>>>>>>> GANE <<<<<<<<<"
270 IF A=2 THEN PRINT "[[[[[[[[[ GANO ]]]]]]]]]"
280 GOTO 10000
300 REM ** TRES EN RAYA **
310 FOR I=1 TO 3
320 IF (T(1,I)=T(2,I)) AND (T(1,I)=T(3,I)) THEN R3=T(1,I):RETURN
330 NEXT I
340 FOR I=1 TO 3
350 IF (T(I,1)=T(I,2)) AND (T(I,1)=T(I,3)) THEN R3=T(I,1):RETURN
360 NEXT I
370 IF (T(1,1)=T(2,2)) AND (T(1,1)=T(3,3)) THEN R3=T(1,1):RETURN
380 IF (T(1,3)=T(2,2)) AND (T(1,3)=T(3,1)) THEN R3=T(1,1):RETURN
390 R3=0:RETURN
500 REM ** LEER JUGADA **
510 INPUT "JUGADA: (FILA, COLUMNA):";F,C
520 RETURN
1000 REM ** GENERAR JUGADA **
1010 F=INT(RND(0)*3+1)
1020 C=INT(RND(0)*3+1)
1030 IF T(F,C)<>0 THEN GOTO 1010
1040 RETURN
10000 END
```

situación buscar tres casillas seguidas y en blanco, y colocar en una de ellas; si esta situación tampoco existe el programa buscará una casilla vacía aleatoriamente. El organigrama correspondiente a estas ideas está desarrollado en la figura 8. El listado correspondiente en la figura 9.

Si el lector introduce el programa en la máquina verá que no es todo lo perfecto que sería de desear, aunque suele hacer tablas no es muy difícil ganarle si se juega con picardía (empezar dos líneas a la vez como muestra la figura 10). Parte de su inexperiencia se debe al hecho de que no se le han metido todas las combinaciones ganadoras; además si se le deja que empiece a jugar él veremos que coge la casilla (1,1) (arriba a la izquierda), en vez de coger la central que es la más interesante en este juego. Por tanto hemos de distinguir qué casillas son más interesantes que otras, pero quizás más importante es su corta capacidad de ver jugadas con anticipación.

Los niveles de pensamiento

Los niveles de pensamiento es lo que en los programas profesionales se llama "nivel de juego" que suele variar de uno a un número que varía de cuatro a ocho, según la máquina. Con este nombre se denomina al proceso en el que la máquina calcula no ya su mejor jugada siguiente, sino las siguientes posibles y elige la que mejor resultado final le da. Como todos los jugadores de ajedrez saben, muchas veces es necesario perder una pieza para poder ganar el juego, siendo esta regla válida para todos los demás juegos. En el programa en su estado actual el ordenador no sabe calcular estas jugadas y juega a ganar el siguiente movimiento, pero no a

Figura 9

```
5 REM ** INICIO **
6 DIM T(3,3), TP(3,3)
10 PRINT "♫"
15 REM ** LEER PRIMER JUGADOR **
20 INPUT "(1=MAQUINA, 2=HUMANO):";A
40 IF (A<>1) AND (A<>2) THEN GOTO 10
45 REM ** ELEGIR TURNO **
50 IF A=2 THEN GOSUB 500
60 IF A=1 THEN GOSUB 1000
65 REM ** COLOCAR FICHA **
70 T(F,C)=A
75 REM ** IMPRIMIR TABLERO **
76 PRINT F,C
77 PRINT "   123"
80 FOR I=1 TO 3
85 PRINT I;
90 FOR J=1 TO 3
100 IF T(I,J)=1 THEN PRINT "O";: GOTO 130
110 IF T(I,J)=2 THEN PRINT "X";:GOTO 130
120 PRINT " ";
130 NEXT J
140 PRINT
150 NEXT I
160 GOSUB 300
170 IF R3<>0 THEN GOSUB 250
180 REM ** TABLERO LLENO? **
190 FOR I=1 TO 3: FOR J=1 TO 3
200 IF T(I,J)=0 GOTO 230
210 NEXT J:NEXT I
220 REM ** FICHERO LLENO **
225 PRINT "TABLAS ***********":GOTO 10000
230 REM ** CAMBIAR TURNO **
240 IF A=1 THEN A=2: GOTO 45
245 A=1 :GOTO 45
250 REM ** TRES EN RAYA **
260 IF A=1 THEN PRINT ">>>>>>>>>> GANE <<<<<<<<<"
270 IF A=2 THEN PRINT "[[[[[[[[[[ GANO ]]]]]]]]]"
280 GOTO 10000
300 REM ** TRES EN RAYA **
310 FOR I=1 TO 3
320 IF (T(1,I)=T(2,I)) AND (T(1,I)=T(3,I)) THEN R3=T(1,I):RETURN
330 NEXT I
```

ganar la partida. La solución ideal consistiría en que el programa calculase todos los juegos posibles y eligiese el mejor. Pero esto revierte varios problemas. En primer lugar calculemos cuantas situaciones posibles de juego hay: en el primer turno, el jugador puede colocar en nueve casillas disponibles, el siguiente jugador podrá colocar en una de las ocho restantes, etc. Las posibilidades de juegos distintos son 9 * 8 * ... * 1. O expresado matemáticamente: 9! (factorial de 9) que nos da el resultado de 362.880 jugadas distintas (si no se lo cree haga el cálculo). En realidad este número es ligeramente inferior, ya que hemos calculado que siempre llenamos el tablero y cuando un jugador gana, el juego se termina. Aun así el número de jugadas disponibles es lo suficientemente grande como para que el ordenador se esté un buen tiempo pensando si además queremos guardar todas las jugadas nos encontraremos con la necesidad de una memoria "de elefante".

Otro problema reside en el hecho de que una vez elegida nuestra jugada no siempre acaba como lo habíamos pensado (para algo tenemos un oponente con libertad de elección), así pues después de cada jugada nuestra hay que ver todas las posibles respuestas del contrario. Si estos dos problemas no le han asustado bastante, piense en este planteamiento realizado para un programa de damas (por ejemplo). En el primer movimiento hay siete posibilidades distintas que se multiplican por otras siete que tiene nuestro adversario, de momento 49 distintas, en nuestro segundo turno podremos mover un máximo de nueve y nuestro adversario lo mismo (y ya van 3.969 juegos distintos), en este caso con cada jugada aumentan las posibilidades de juegos

```
340 FOR I=1 TO 3
350 IF (T(I,1)=T(I,2)) AND (T(I,1)=T(I,3)) THEN R3=T(I,1):RETURN
360 NEXT I
370 IF (T(1,1)=T(2,2)) AND (T(1,1)=T(3,3)) THEN R3=T(1,1):RETURN
380 IF (T(1,3)=T(2,2)) AND (T(1,3)=T(3,1)) THEN R3=T(1,1):RETURN
390 R3=0:RETURN
500 REM ** LEER JUGADA **
510 INPUT "JUGADA: (FILA, COLUMNA):";F,C
520 RETURN
1000 REM ** GENERAR JUGADA **
1010 REM ** GENERAR TABLERO DE PRUEBA **
1020 FOR I=1 TO 3: FOR J = 1 TO 3
1030 TP(I,J)=T(I,J)
1040 NEXT J, I
1050 REM ** GENERAR VALORES **
1055 FOR I=1 TO 3: SH(I)=0:SV(I)=0:FH(I)=0:FV(I)=0:  NEXT I: D1=0:D2=0
1057 F1=0:F2=0
1060 FOR I=1 TO 3: FOR J=1 TO 3
1070 IF TP(J,I)=1 THEN SV(I)=SV(I)+1
1080 IF TP(J,I)=2 THEN SV(I)=SV(I)-1:FV(I)=1
1090 IF TP(I,J)=1 THEN SH(I)=SH(I)+1
1100 IF TP(I,J)=2 THEN SH(I)=SH(I)-1:FH(I)=1
1110 NEXT J
1120 NEXT I
1130 FOR I=1 TO 3: IF TP(I,I)=1 THEN D1=D1+1
1140 IF TP(I,4-I)=1 THEN D2=D2+1
1150 IF TP(I,I)=2 THEN D1=D1-1:F1=1
1160 IF TP(I,4-I)=2 THEN D2=D2-1:F2=1
1170 NEXT I
1180 REM ** BUSCAR DOS EN RAYA **
1190 IF D1=2 THEN TT$="D":NU=2: GOTO 3000
1200 IF D1=2 THEN TT$="D":NU=1: GOTO 3000
1210 FOR I=1 TO 3
1220 IF SV(I)= 2 THEN TT$="C":NU=I:GOTO 3000
1230 IF SH(I)= 2 THEN TT$="F":NU=I:GOTO 3000
1240 NEXT I
1250 REM ** BUSCAR 2 CONTRARIAS EN RAYA **
1260 IF D2=-2 THEN TT$="D":NU=2: GOTO 3000
1270 IF D1=-2 THEN TT$="D":NU=1: GOTO 3000
1280 FOR I=1 TO 3
1290 IF SV(I)=-2 THEN TT$="C":NU=I:GOTO 3000
1300 IF SH(I)=-2 THEN TT$="F":NU=I:GOTO 3000
```

distintos. La cifra final no existe, ya que se puede dar el caso de partidas que no acaben nunca (ambos jugadores con una sola dama y en determinadas casillas), poniéndole un límite racional al juego (determinadas situaciones se consideran tablas) seguimos teniendo posibilidades suficientes como para llenar la memoria del ordenador más potente y tardaría lo suficiente como para que lo viesen acabar nuestros sucesores lejanos). Una vez comprobada la imposibilidad del programa que piense todas las jugadas nos debemos conformar con un método que nos deje calcular las próximas jugadas (con siete es más que suficiente) y además debemos ponerle ciertas restricciones o nos seguirá saliendo un programa que tarda años en pensar (para no hablar de la memoria necesaria). Estas restricciones se refieren a que se deben descartar en principio las jugadas demasiado descabelladas (por ejemplo ceder la reina en el ajedrez sin recibir nada a cambio). Dependiendo de las restricciones que le pongamos, el programa jugará mejor o peor y será más o menos rápido.

El segundo problema reside en el hecho de que el oponente no suele hacer lo que nosotros queremos que haga y, por tanto, destroza nuestra jugada cuidadosamente planeada. Para evitar este hecho, realizamos una búsqueda del mínimo: A cada posición posible del juego le asignamos una puntuación: si hay tres fichas nuestras en raya le podemos asignar un 100 a esa posición, si hay dos y la tercera está en blanco, le podemos asignar un 50; si hay dos fichas contrarias en línea y la tercera está en blanco, le asignamos un —50, etc. De modo que las buenas posiciones nuestras suman puntos y las buenas del contrario los restan. Asi-

```
1305 NEXT I
1310 REM ** 1 NUESTRA Y DOS VACIAS **
1320 IF (D1=1) AND (F1=0) THEN TT$="D":NU=1:GOTO 3000
1330 IF (D2=1) AND (F2=0) THEN TT$="D":NU=2:GOTO 3000
1340 FOR I=1 TO 3
1350 IF (SV(I)= 1)AND (FV(I)=0) THEN TT$="C":NU=I:GOTO 3000
1360 IF (SH(I)= 1)AND (FH(I)=0) THEN TT$="F":NU=I:GOTO 3000
1370 NEXT I
1380 REM ** 3 VACIAS **
1390 IF((D1=0) AND (F1=0))THEN TT$="D":NU=1:GOTO 3000
1400 IF (D2=0) AND (F2=0) THEN TT$="D":NU=2:GOTO 3000
1410 FOR I=1 TO 3
1420 IF (SV(I)= 0)AND (FV(I)=0) THEN TT$="C":NU=I:GOTO 3000
1430 IF (SH(I)= 0)AND (FH(I)=0) THEN TT$="F":NU=I:GOTO 3000
1440 NEXT I
1450 REM ** ALEATORIO **
1460 F=INT(RND(1)*3+1)
1470 C=INT(RND(1)*3+1)
1480 IF TP(F,C)<>0 THEN GOTO 1450
1490 RETURN
3000 REM ** COGE CASILLA **
3010 IF TT$<>"D" THEN GOTO 3100
3020 FOR I=1 TO 3
3030 IF (TP(I,I)=0) AND (NU=1)THEN F=I:C=I:GOTO 3500
3035 IF (TP(I,4-I)=0) AND (NU=2)THEN F=I:C=4-I:GOTO 3500
3040 NEXT I
3050 STOP: REM ** ERROR **
3100 IF TT$<>"F" THEN GOTO 3200
3110 FOR I=1 TO 3
3120 IF TP(NU,I)=0 THEN F=NU:C=I:GOTO 3500
3130 NEXT I
3140 STOP: REM ** ERROR **
3200 IF TT$<>"C" THEN GOTO 3300
3210 FOR I= 1 TO 3
3220 IF TP(I,NU)=0 THEN F=I:C=NU:GOTO 3500
3230 NEXT I
3240 STOP: REM ** ERROR **
3300 STOP: REM ** ERROR **
3500 RETURN
10000 END
```

mismo a cada casilla se le debe dar una puntuación, si en la casilla central hay una ficha nuestra sumaremos diez puntos, si hay una del enemigo restaremos diez. De este modo cada posición distinta del juego tendrá una puntuación que nos indica lo buena que puede ser. Una vez visto cómo valoramos las jugadas realizamos una búsqueda por árbol (ver figura 11). En esta búsqueda realizamos todas las jugadas posibles con el nivel de profundidad elegido (en el ejemplo es dos) y les damos una puntuación; una vez hecho esto no buscamos la jugada con más valor, sino que en cada intersección en la que movamos nosotros (A, B, C,... I) tomamos la menor puntuación que se haya obtenido y se la asignamos a esa intersección. En el ejemplo de la figura 12 la intersección A genera un valor 100, otro de 50 y otro de —20, al nudo A le asignaremos el valor —20, considerando que nuestro ponente es perfecto y nos va a contestar siempre con la mejor jugada (aunque no totalmente cierto, es mejor pensar eso a que es tonto y juega a perder) en la intersec-

Figura 6 A

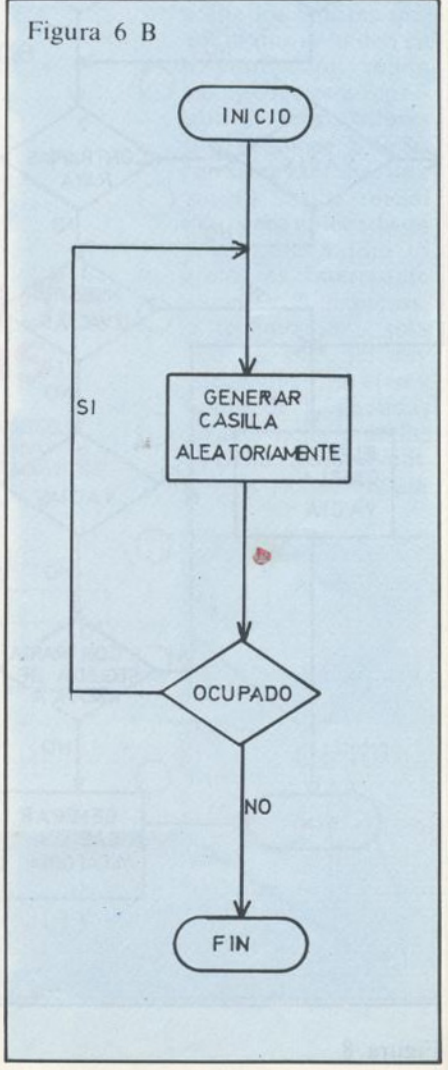


Figura 6 B

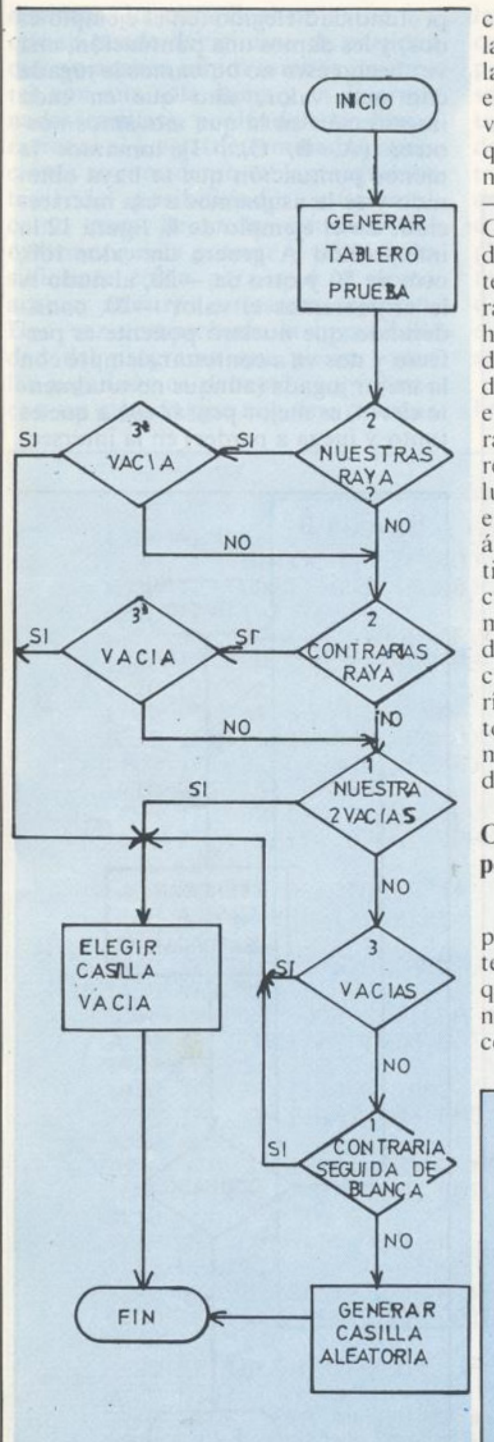


Figura 8

ción B el menor valor es de —50 y en la C es de —10. Por tanto elegiríamos la intersección C, ya que es la que, en el peor caso, perdemos menos. Una vez establecida esta idea, pensemos que después de explorar la rama A nos sale una puntuación menor de —30. Al empezar a explorar la B y la C nos salen puntuaciones de —50 y de —100 (figura 13), en ese caso no tendremos que seguir explorando esas ramas, ya que la puntuación que nos ha salido de principio es peor que la de la rama A, con lo que esta rama la desecharemos. Si al seguir adelante encontramos que al explorar toda la rama B su puntuación es —10, elegiremos ésta como la rama preferida en lugar de la A. Así no necesitamos explorar todas las ramas posibles del árbol con lo que ahorramos mucho tiempo y sólo necesitamos guardar cuál es la rama que nos ha dado la mejor puntuación mínima. Si en vez de pensar con dos jugadas la exploración la hiciésemos con cuatro, deberíamos seguir el mismo procedimiento, pero cada posición la evaluaríamos al final de las cuatro jugadas y no de las dos.

Optimizando los niveles de pensamiento

Aunque los algoritmos dados le permiten realizar programas pensantes, se pueden perfeccionar de modo que piensen mejor y más rápido. Si nos fijamos en la figura 14, veremos como la puntuación menor de la rama A es de cinco, mientras que la primera que hemos examinado de la rama B nos sale con una puntuación de tres, por lo tanto no necesitamos seguir examinando esta rama para saber que es peor que la A. Vemos que podemos eliminar una rama cuando veamos que su puntuación es inferior a la de la rama elegida. Realizando esta operación con todas las ramas conseguiremos gran aumento de velocidad. A continuación consideremos que aplicando este procedimiento en una jugada nos sale que la mejor rama es la última que examinamos, evidentemente hemos perdido un tiempo inútil realizando búsquedas exhaustivas hasta encontrar la mejor jugada. Para arreglar esto conviene ordenarlas desde un principio con un orden lógico que (con bastante probabilidad) nos de la mejor jugada al principio. Por ejemplo en el juego del TIC-TAC-TOE consideremos primero la posibilidad de la casilla central (de principio la más apetitosa), luego miraremos las cuatro esquinas y al final miraremos las cuatro casillas restantes, lo más probable es que en una de las primeras que hemos mirado esté la jugada que buscamos. Otro ejemplo está en las aperturas de ajedrez. Las jugadas más usadas (y con razón) es empezar abriendo por el centro, el peón de rey o de reina, y desarrollar la apertura en base a esas jugadas. De nada nos servirá empezar examinando los peones laterales o las torres y alfiles que no se pueden mover.

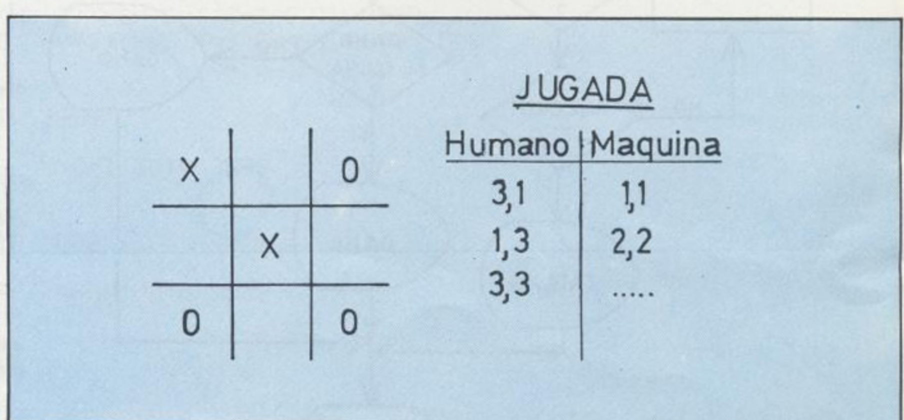


Figura 10

Variación de algoritmos

Como hemos visto en el apartado anterior, mover un peón lateral en el ajedrez al principio del juego no suele proporcionar buenos resultados, pero esta situación no es igual cuando el juego se halla totalmehte desarrollado o al final, en ese caso quizás la mejor jugada sea mover un peón lateral. Así pues, nuestro algoritmo debe variar según se encuentre el juego. Para averiguar esto último se puede usar una gran cantidad de algoritmos, pero se debe tener cuidado en que no sea fácil engañarle. Supongamos que en el ajedrez le decimos al programa que cambie de estrategia cuando el contrario mueva

lo más genérica posible, ver cuántas piezas están fuera de su casilla original, cuántas piezas han sido comidas, etc. Estos algoritmos puede que no sean demasiado precisos, pero son más difíciles de engañar.

Funciones de evaluación

Como se ha visto anteriormente, se deben dar valores a las situaciones para saber cómo está la partida y qué debemos hacer. Esta tarea tan sencilla para los humanos es realmente difícil de realizar en un ordenador, debido a su falta de visión de conjunto del juego. Esta falta la debemos reemplazar con un buen algoritmo que le haga valorar la situación. En el juego

debe utilizar en ocasiones especiales, ya que si nos comen al rey perdemos la partida, su uso se debe limitar a ver el interés de colocar el rey en una casilla determinada. Estas puntuaciones no son así necesariamente; uno puede cambiarlas hasta que vea que el programa juega mejor. Otra cuestión a considerar son las casillas, como vimos en el juego del TIC-TAC-TOE la casilla central es la mejor seguida de las esquinas. En las aperturas de ajedrez se considera de vital importancia controlar el centro del tablero, etc. Así pues, debemos asignar una puntuación distinta a cada casilla o, dicho de otro modo, cada pieza valdrá distinto según la casilla donde se encuentre. Debemos tener una matriz que almacene los valores que multiplican a la pieza que se hallen en

puntuación válida ajedrez está repre- ...a 15. Estos valores al valor de la ficha ...n su casilla. Así una ...n su casilla inicial ...os, pero colocada en centrales valdrá 15 ...mo, es interesante ...ciaciones de piezas. tenemos una sola en la fila del rey se encuentra el rey ...cipio de la partida) ...e una ficha muerta; ...a ficha está apoyada ... más fichas en esa

Figura 11

Figura 12

Variación de algoritmos

Como hemos visto en el apartado anterior, mover un peón lateral en el ajedrez al principio del juego no suele proporcionar buenos resultados, pero esta situación no es igual cuando el juego se halla totalmente desarrollado o al final, en ese caso quizás la mejor jugada sea mover un peón lateral. Así pues, nuestro algoritmo debe variar según se encuentre el juego. Para averiguar esto último se puede usar una gran cantidad de algoritmos, pero se debe tener cuidado en que no sea fácil engañarle. Supongamos que en el ajedrez le decimos al programa que cambie de estrategia cuando el contrario mueva el rey. Si la persona que juega contra el ordenador descubre el truco intentará mover el rey lo antes posible con lo que el ordenador se creerá que nos hallamos en mitad de la partida y jugará de acuerdo con esas suposiciones destrozando su defensa y dándonos la victoria. La valoración que se haga para cambiar de táctica debe ser lo más genérica posible, ver cuántas piezas están fuera de su casilla original, cuántas piezas han sido comidas, etc. Estos algoritmos puede que no sean demasiado precisos, pero son más difíciles de engañar.

Funciones de evaluación

Como se ha visto anteriormente, se deben dar valores a las situaciones para saber cómo está la partida y qué debemos hacer. Esta tarea tan sencilla para los humanos es realmente difícil de realizar en un ordenador, debido a su falta de visión de conjunto del juego. Esta falta la debemos reemplazar con un buen algoritmo que le haga valorar la situación. En el juego del TIC-TAC-TOE todas las fichas valen lo mismo, mientras que en ajedrez al existir fichas distintas cada una puede tener un valor. Una valoración puede ser: Peón 1 punto, alfil 3 puntos, caballo 3 y medio, torre 5 puntos, reina 10 puntos y rey 4 puntos; esta puntuación del rey se debe utilizar en ocasiones especiales, ya que si nos comen al rey perdemos la partida, su uso se debe limitar a ver el interés de colocar el rey en una casilla determinada. Estas puntuaciones no son así necesariamente; uno puede cambiarlas hasta que vea que el programa juega mejor. Otra cuestión a considerar son las casillas, como vimos en el juego del TIC-TAC-TOE la casilla central es la mejor seguida de las esquinas. En las aperturas de ajedrez se considera de vital importancia controlar el centro del tablero, etc. Así pues, debemos asignar una puntuación distinta a cada casilla o, dicho de otro modo, cada pieza valdrá distinto según la casilla donde se encuentre. Debemos tener una matriz que almacene los valores que multiplican a la pieza que se hallen en esa casilla. Una puntuación válida para un tablero de ajedrez está representada en la figura 15. Estos valores deben multiplicar al valor de la ficha que se encuentra en su casilla. Así una torre colocada en su casilla inicial valdrá cinco puntos, pero colocada en una de las cuatro centrales valdrá 15 puntos. Por último, es interesante considerar las asociaciones de piezas. En el ajedrez si tenemos una sola pieza colocada en la fila del rey enemigo (donde se encuentra el rey enemigo al principio de la partida) suele considerarse una ficha muerta; o casi, pero si esta ficha está apoyada por otras o hay más fichas en esa

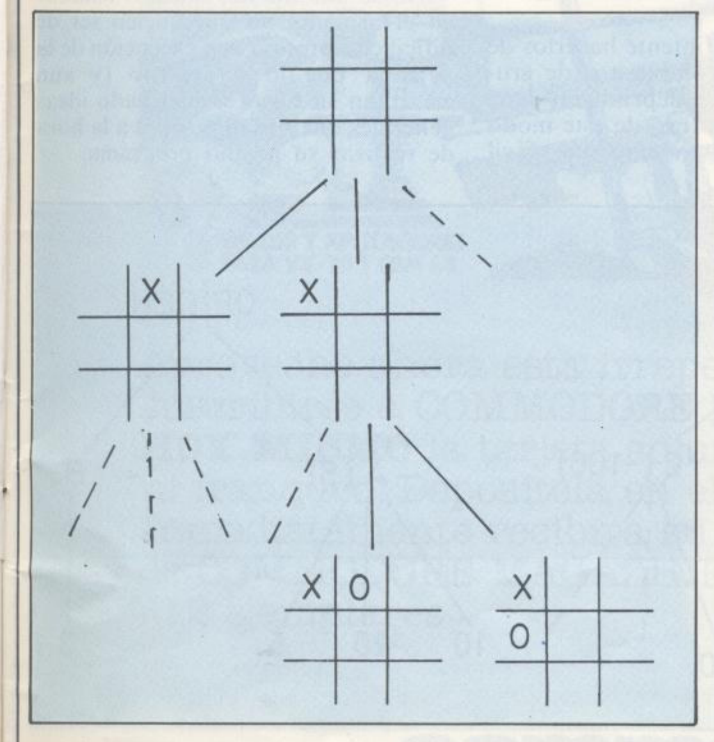


Figura 11

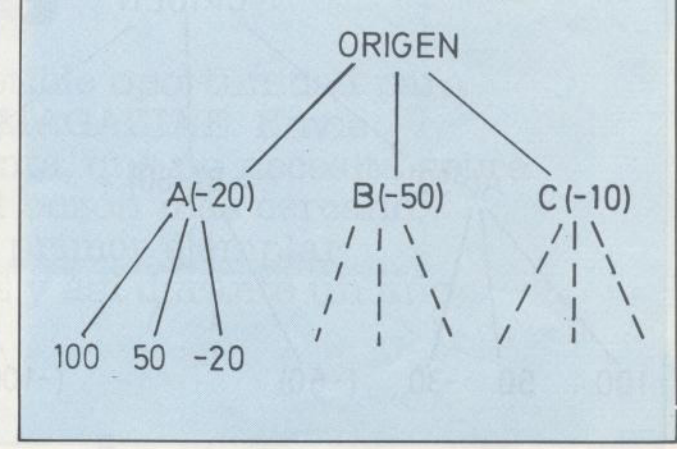
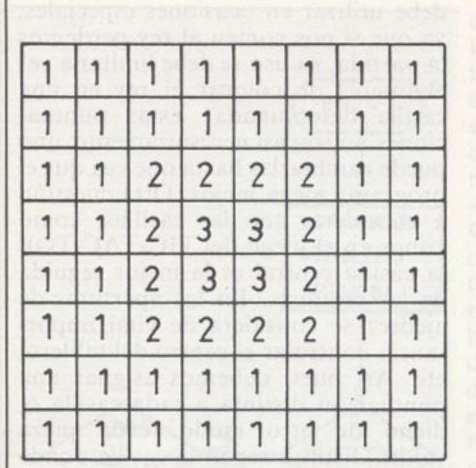
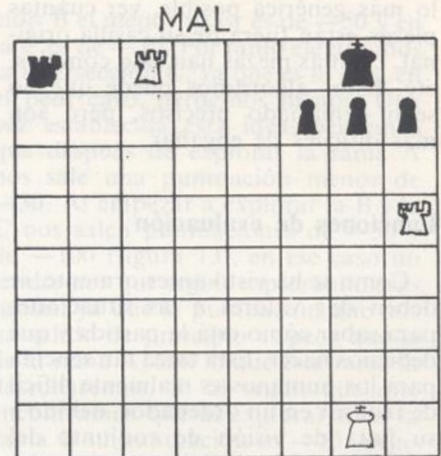


Figura 12



1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	2	2	2	2	1	1
1	1	2	3	3	2	1	1
1	1	2	3	3	2	1	1
1	1	2	2	2	2	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

Figura 15



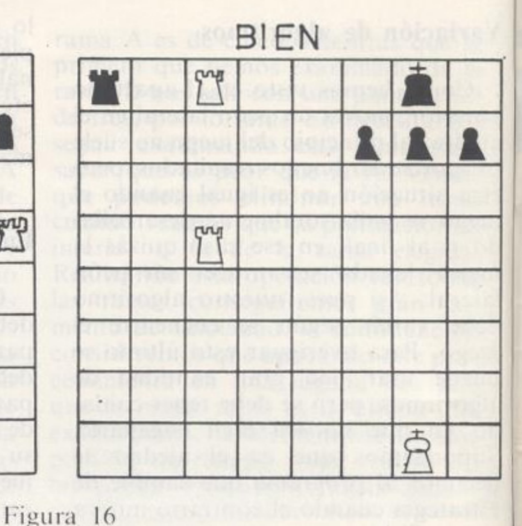


Figura 16

misma línea su situación cambia y su puntuación debe ser muy positiva, ya que está inflingiendo un gran daño al contrario (figura 16).

Consideraciones finales de los juegos inteligentes

Cuanso se ponga a realizar un programa para jugar a este tipo de juegos no empiece diseñando el tablero y las fichas para dejarlo precioso y luego realizar el algoritmo de juego. Primero consiga que el programa juegue. Las jugadas no las tiene que expresar de un modo muy bonito o espectacular (por lo menos, al principio) bastará con que diga a la casilla qué ha movido. Una vez realizado un algoritmo de juego que funcione preocúpese por dejarlo bonito, pero no antes.

Los programas intente hacerlos de forma modular, compuestos de grupos de subrutinas independientes que se llamen unas a otras, de este modo si alguna falla es mucho más fácil modificarla sin tener que tocar las demás.

No intente hacer "el mejor programa de ajedrez del mundo" como su primer programa de juego. Empiece por cosas sencillas y según las vaya dominando pase a realizar juegos más complicados.

No se han metido muchos listados en el artículo, ya que suelen ser de difícil compresión con excepción de la persona que lo ha escrito (y aun así...), en su lugar se han dado ideas generales que son más útiles a la hora de realizar su propio programa.

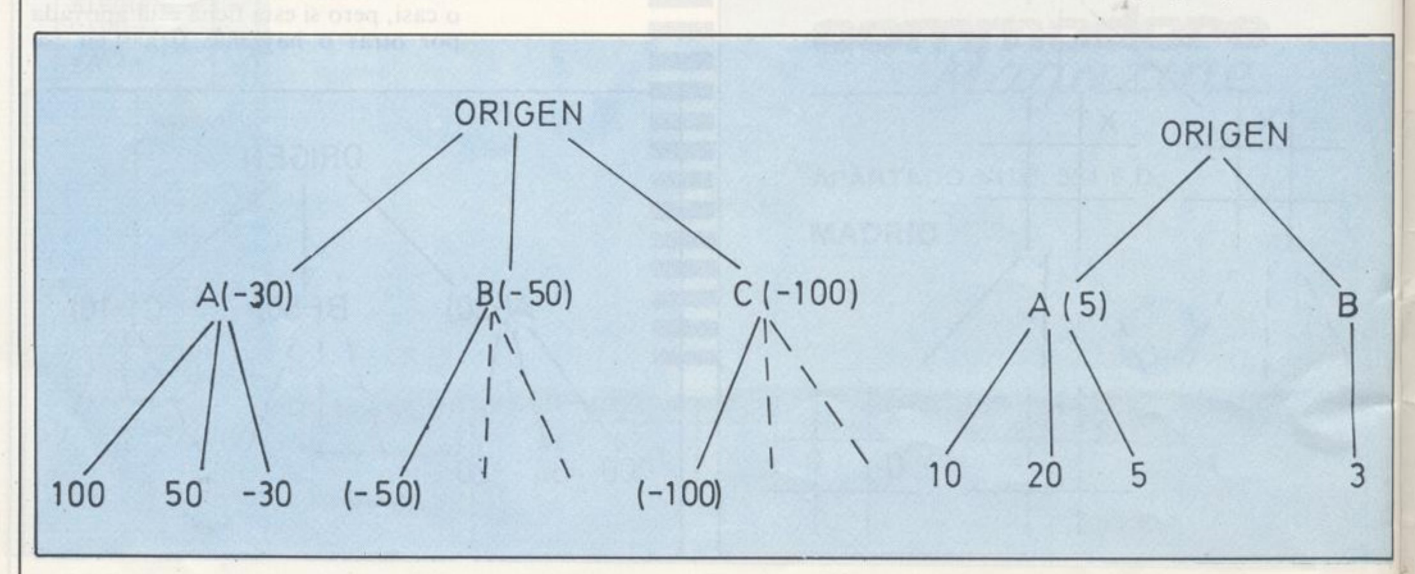


Figura 13

Figura 14

La revista imprescindible para todo el usuario de
Ordenadores COMMODORE

commodore *Magazine*

commodore Magazine
Año I - Número 1 - 200 ptas.
CALC RESULT, A FIN DE CUENTAS
Más potencia con Victree
Cómo adaptar cualquier cassette
JUEGOS Y APLICACIONES PARA VIC-20 Y CBM 64

OFERTA ESPECIAL DE INTRODUCCIÓN

Aproveche ahora esta irrepetible oportunidad para suscribirse a COMMODORE MAGAZINE. Envíe **HOY MISMO** la tarjeta adjunta, que no necesita sobre ni franqueo. Depositela en el buzón más cercano. Inmediatamente recibirá su primer ejemplar de COMMODORE MAGAZINE y así durante un año (12 ejemplares).



Bravo Murillo, 377.
Tel. 733 96 62
Madrid - 20

Iniciación al lenguaje má

INSTRUCCIONES Y MODOS DE DIRECCIONAMIENTO DEL 6502

En los capítulos anteriores hemos visto la estructura básica que conforma un sistema basado en microprocesador, cada una de sus partes y los caminos de unión entre ellas, pero ya es el momento de hablar de las instrucciones que puede ejecutar nuestro sistema. A lo largo de estos capítulos nos estamos refiriendo concretamente a la familia **6500** y en particular a la **CPU 6502**. En el primer capítulo, cuando hablamos de las características fundamentales decíamos que tenía un repertorio de 56 instrucciones, pues bien veamos qué significa esto. Una instrucción consiste en la posibilidad de realizar una tarea determinada por parte de la CPU; por ejemplo, sacar un dato de memoria, almacenar un dato en un registro temporal, etc. Recordemos que llamábamos registro temporal a unas posiciones determinadas (a modo de casilleros donde se guardaban los unos y ceros) destinados a almacenar cualquier dato durante un tiempo determinado y luego poder disponer de él nuevamente.

A continuación nos detendremos en cada una de estas 56 instrucciones, detallando su simbología, la forma en la que se ejecuta la instrucción y el modo de direccionamiento usado en cada una, así como los ejemplos que sirvan para esclarecer cualquiera de las instrucciones comencemos por la ADC y sigamos en orden alfabético:

ADC: Suma memoria al acumulador con acarreo. Mediante esta instrucción sumamos un valor contenido en memoria con el acarreo procedente de otras operaciones anteriores, almacenando el resultado obtenido, de nuevo en el acumulador.

La representación simbólica es: $(A)+(M)+C \rightarrow A$

Donde A representa el contenido del acumulador, M el contenido de una posición de memoria, que se especifica en la instrucción, y C el acarreo procedente de la operación anterior.

La flecha indica que el resultado de la suma de nuevo irá al acumulador.

Acarreo viene a significar lo mismo que cuando en una operación decimos "... y me llevo uno". Esta es una instrucción de las llamadas del grupo "cero", y por tanto, tiene los modos de direccionamiento siguientes: inmediato; absoluto; a página cero, absoluto en X; absoluto en Y, indexado indirecto e indirecto indexado.

AND: Realiza la función lógica AND (Multiplicación lógica) entre el contenido de la posición de memoria especificada y el contenido del acumulador, volviendo a guardarse el resultado obtenido en el acumulador. Esta función lógica se realiza bit a bit.

Ejemplo: AND $(A) \cdot (M) \rightarrow A$ ó $A \wedge M \rightarrow A$

Esta instrucción también pertenece al grupo cero, por tanto, tiene todos los modos de direccionamiento.

En el ejemplo anterior el código en hexadecimal correspondiente al direccionamiento absoluto en X es 3D, el cual necesitá cuatro ciclos de máquina y consta de tres *bytes* o palabras de 8 bits.

MODOS DE DIRECCIONAMIENTO

Dependiendo del camino que recorra una instrucción para definir o situar al operando, variará el código máquina asociado a dicha instrucción. Esto supone que una misma instrucción puede tener distintos códigos máquina, dependiendo de la forma de direccionamiento del operando.

Muchas de las instrucciones que puede ejecutar un ordenador, son instrucciones lógicas y aritméticas, que como dijimos en el capítulo anterior se realizan mediante la ALU. En la ejecución de este tipo de operaciones, participan normalmente dos operandos uno de ellos siempre está en el acumulador, con lo cual sólo necesitamos definir el otro operando necesario para realizar la operación. Este segundo operando lo podeos definir directamente en la instrucción. Esta forma de direccionar el operando se llama "Direccionamiento Inmediato". En otro tipo de direccionamiento, el operando está contenido en una posición de memoria determinada. Para ello tenemos que indicar, justo detrás del código de operación, los *bytes* que indican su dirección. Esta forma de direccionar el operando se llama "Direccionamiento Directo".

Un caso particular de direccionamiento directo es el de definir únicamente la posición de memoria, pero para ello es necesario hacer referencia a la página en la que está localizada esa posición de memoria. Si nos referimos, por ejemplo, a la página "cero", al direccionamiento se le co-

Familia 6500

quina

ASL: Desplazamiento aritmético a la izquierda. Mediante esta operación se desplaza todo el *byte* un bit hacia la izquierda, poniendo un "cero" en el bit de menor peso del contenido de una posición de memoria o del acumulador. Esta instrucción modifica los indicadores N, Z, y C del registro de estado.

ASL C ← | B7 | ..... | Bϕ | ← ϕ

ASL es una instrucción del tipo leer/modificar/escribir y tiene los modos de direccionamiento siguientes: acumulador; página cero; página cero en X; absoluto; absoluto en X.

BCC: Salto si el acarreo está desactivado. Salto si C = O. Esta instrucción comprueba el estado del bit de acarreo y realiza un salto condicional si el bit de acarreo está a "cero". No afecta ni a los indicadores ni a otros registros. El modo de direccionamiento es relativo.

BCS: Salto si el acarreo está activado. Salto si C = 1. Es igual al anterior excepto que C = 1.

BEQ: Salto si el resultado de una operación es cero. Salto si Z = 1. No afecta a ningún *flag* ni registro, salvo en caso de que Z = 1 modificando entonces el contador de programa. El modo de direccionamiento es relativo.

BIT: Comprueba los bits de memoria con el acumulador. Operación: A ∧ M, M7 → N, M6 → V (∧ = AND)

Los bit 6 y 7 son transferidos al registro de estado. Si el resultado de A ∧ M es cero, entonces Z = 0.

Los modos de direccionamiento son: página cero y absoluto. Ejemplo: Porción de programa empleando la instrucción BIT.

- LDA: Cargar MASK en el acumulador:
- MASK (máscara).
- BIT: Examina el primer valor de la memoria por medio del bit de la máscara.
- ADL1.
- ADH1.
- BNE: Salta si se activa. +50.
- BIT: Examina el segundo valor de la memoria por medio del bit de la máscara, etcétera.

BMI: Salto si el resultado es negativo. Operación: salto si N = 1. El salto si el resultado es negativo, se utiliza para determinar si el resultado anterior fue negativo o el bit 7 estaba activado. No afecta a los indicadores y el direccionamiento es relativo.

BNE: Salto si el resultado es distinto de cero. Salto si Z = O. A esta instrucción se le suele llamar "salto en desigualdad". La instrucción examina el indicador Z y realiza el salto condicional si el indicador Z está a "cero". Como en la instrucción anterior tampoco afecta indicadores y su direccionamiento es relativo.

BPL: Salto si el resultado es positivo. Salto si N = 0. Esta instrucción es la complementaria al salto si el resultado es negativo. El salto es condicional y se realiza cuando el bit "N" del registro de indicadores es cero. No

noce con el nombre de "Direccinamiento en página cero".

En la familia **6500** existen además otros tipos de direccionamientos, como veremos a continuación.

Detrás del código de operación se coloca un valor determinado, el cual hay que añadir al contenido de uno de los registros índices X o Y (de la unidad central de proceso) con el fin de averiguar la dirección de memoria donde radica el operando. Se dice entonces que el direccionamiento es INDEXADO. Si por el contrario colocamos un valor detrás del código de operación, que se suma al contador de programa, el direccionamiento será RELATIVO.

Otra de las formas posibles de direccionamiento, es el llamado direccionamiento INDIRECTO. Se da cuando ponemos detrás del código de operación una dirección de memoria, en cuyo contenido y el de la siguiente posición se encuentra la dirección donde se localiza el operando.

Todos los microprocesadores de la familia **6500** disponen de 13 modos de direccionamiento posibles.

Según sea el modo de direccionamiento, las instrucciones pueden estar formadas por 1, 2 ó 3 *bytes* (8 bits), pero siempre el primero de estos *bytes* corresponde al código máquina de la instrucción.

Direccionamiento Inmediato

Las instrucciones con este tipo de direccionamiento, tienen dos *bytes.*

El primero de ellos representa el código de operación y además de información sobre el modo de direccionamiento.

El segundo *byte* queda definido por el programador según las necesidades

afecta indicadores y el direccionamiento es relativo.

BRK: Comando de parada.

Operación: Interrupción forzada PC + 2 ↓ P ↓

Este comando obliga al microprocesador a ir a una rutina de interrupción que está bajo el control del programa.

El contador de programa del segundo *byte*, después de la instrucción BRK, se almacena automáticamente en la "pila", junto con el estado del microprocesador al principio de la instrucción de parada. La instrucción no afecta a los indicadores de *flags* y su direccionamiento es en modo implícito. Este vector de interrupciones se sitúa en las posiciones FFFE y FFFF.

BVC: Saltar si el *overflow* (desbordamiento) está desactivado. La operación es: Salto si V = O.

Mediante esta instrucción comprobamos el estado del indicador V y realiza un salto condicional, si el indicador está desactivado. No afecta indicadores y su modo de direccionamiento es relativo.

BVS: Saltar si el desbordamiento está activado. Salto si V = 1. Esta instrucción es la opuesta a la anterior y su funcionamiento es el mismo, salvo que salta si V = 1.

CLC: Desactiva el indicador de acarreo. Pone a 0 la casilla correspondiente en el registro de indicadores: 0 → C. Precede normalmente a la instrucción ADC. No se afectan indicadores ni registros.

CLD: Pone el indicador *(flag)* de modo decimal (D) a cero. Es decir, desactiva el modo decimal: 0 → D. Todas las operaciones aritméticas que se encuentran a continuación realizan sus operaciones en modo binario. Esta instrucción tampoco afecta al registro de indicadores.

CLI: Borra el indicador de prohibición de interrupciones. Es decir, pone el *flag* de interrupciones (I) a cero: 0 → I. Mediante esta instrucción permitimos que actúe la línea $\overline{IRQ}$.

La instrucción CLI es de un solo *byte* (8 bits) y el modo de su direccionamiento es implícito.

CLV: Pone el indicador *(flag)* de *overflow* a cero: 0 → V. Recordemos que el registro de indicadores o *flags* es:

N	Z	C	I	D	V
					ϕ

En este caso V que es el *flag* indicador de desbordamiento que se pone a cero mediante esta instrucción.

CMP: Compara el contenido de la posición de memoria indicada con el contenido del acumulador. Operación: A-M. La operación se realiza sin modificar el contenido del acumulador y al realizarla se modifican los indicadores acarreo (C), negativo (N) y cero (Z).

Siendo "A" el acumulador y "M" la memoria, el resultado de la comparación puede ser:

	N
(A) < (M)	cualquiera
(A) = (M)	"O"
(A) > (M)	cualquiera

Cuando ponemos algo entre paréntesis, por ejemplo (A), significa "el contenido de A". Por ejemplo: (A) > (M), esto se leería: el contenido del acumulador es mayor que el contenido de la posición de memoria M.

Ejemplo de la instrucción CMP:
- LDA: Cargar valor.
- ADL: Dirección baja.
- ADH: Dirección alta.
- CMP: Comparar contador 1 al acumulador.
- COUNT 1.
- BEQ.

Etcétera. Si es igual hacer...

MODOS DE DIRECCIONAMIENTO

de su programa. Se utiliza en algunas instrucciones de operaciones lógicas y aritméticas, así como en operaciones de carga y comparación.

Es el direccionamiento más sencillo, siendo utilizado por las instrucciones siguientes:

ADC, AND, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA, y SBC.

Direccionamiento absoluto

Las instrucciones que utilizan este tipo de direccionamiento necesitan tres *bytes*. Como ya hemos dicho el primer *byte* es el código de operación e indica el modo de direccionamiento. Esto es común para todos los modos. El segundo *byte* indica la parte baja (mnos significativa) de la dirección del operando y en el tercero la parte alta de esta dirección. Este puede ser el modo de direccionamiento más normal. Se utiliza en las instrucciones: ADC, AND, ASL, BIT, CMP, CPX, CPY, DEC, EOR, INC, JMP, JSR, LDA, LDX, LDY, LSR, ORA, ROL, ROR, SBC, STA, STX, STY.

	Operando
(A) ← (2140)	2140

Direccionamiento en página cero

Ejemplo	Opera
(A) ← (00,20)	(00,2

Las instrucciones son de dos *bytes*, siendo el segundo el que contiene la dirección efectiva de la página cero de la memoria. Este tipo de direccionamiento permite una menor ocupación de memoria y una mayor rapidez en la ejecución. Otra ventaja es que sólo

Familia 6500

Es una instrucción del grupo uno, y por tanto, sus modos de direccionamiento son: inmediato, página cero, página cero en X, absoluto, absoluto en "X", en "Y", etc.

CPX: Compara el registro de índice "X" con la memoria. Operación: X-M. Esta operación realiza la resta entre el contenido de X y el operando aludido bien inmediatamente o bien de acuerdo con el direccionamiento empleado, que puede ser: inmediato, absoluto y página cero. El resultado

C	Z	V
"O"	"O"	No cambia
"1"	"1"	,,
"1"	"O"	,,

de esta operación no se almacena, quedando solamente reflejado el carácter de dicho resultado.

CPY: Es igual a la anterior pero referida al registro "Y". Operación: Y-M.

DEC: Decrementar el contenido de la memoria en 1. Operación (M) - 1 → (M). Esta instrucción resta contenido de la posición de memoria especificada en M y el resultado lo vuelve a almacenar de nuevo en M. Puede modificar los *flags* N y Z. Los modos de direccionamiento son: página cero, página cero en X, absoluto, absoluto en X.

DEX: Decrementar el registro de índice X en uno. Operación: (X) —1 → X. Esta instrucción resta uso al valor que tengamos en el registro X (Registro índice) y almacena el resultado de nuevo en el registro X. La instrucción DEX es de un solo *byte* y su modo de direccionamiento es implícito. Afecta a los indicadores N y Z.

DEY: Decrementa el registro de índice Y en uno. Operación (Y) —1 → Y. En lo demás es idéntica en todo a la instrucción anterior.

EOR: Realiza la operación lógica Or-exclusiva, entre el contenido especificado de la memoria y el acumulador. (A) ⊻ (M) → A. ⊻ significa Or exclusiva. Esta operación se realiza bit a bit y el resultado se vuelve a almacenar en el acumulador. También se suele representar como:

$(A) \oplus (M) \rightarrow A$

Es una instrucción del grupo un (ver los modos de direccionamiento de este grupo). Uno de los posibles usos de esta instrucción es complementar *bytes*.

Ejemplo:

LDA	1010	1111
EOR	1111	1111
STA	0101	0000

INC: Incrementar el contenido de la memoria en uno. Operación: (M) + 1 → M. Esta operación suma "1" al contenido de la posición de memoria direccionada. Puede modificar los indicadores N y Z. Los modos de direccionamiento de esta instrucción son: página cero, página cero en X, absoluto y absoluto en X.

INX: Incrementa el registro de índice X en uno. Operación: (X) + 1 → X. Esta instrucción suma "uno" al valor presente en el registro X. Es un incremento de 8 bits que no afecta a la operación de acarreo. Afecta a los indicadores X y Z. El modo de direccionamiento es implícito.

INY: Incrementa el registro de índice Y en uno. Operación: (Y) + 1 → Y. Esta instrucción es similar a la anterior, pero con el registro Y.

JMP: Salto a una nueva posición. Operación:

(PC + 1) → PCL
(PC + 2) → PCH

Mediante esta instrucción se rompe la secuencia del programa saltando incondicionalmente al lugar de memoria especificado en el operando. No afecta a los indicadores, pero sí al contador del programa. Es una instrucción de tres *bytes*, donde el segundo y el tercero representan los *bytes* bajo y alto respectivamente de

Código de op.

LDA 2140 AD

invierte tres ciclos máquina (ciclos de reloj) para ejecutarse.

ndo **Código de op.**

0) A5

Direccionamiento Indexado en página cero

Salvo en LDX y STX que pueden modificarse por el registro Y, el modo de direccionamiento en página cero, sólo puede ser Indexado por medio del registro "X". Las instrucciones que tienen este tipo de direccionamiento son: ADC, AHD, ASL, CMP, DEC, EOR, INC, LDA, LDY, LSR, ORA, ROL, ROR, SBC, STA, STY.

Ejemplo:	Operando	Código OP.
(A) ← (00,88 + x)	(00,88 + x)	LDA 88, x B5

Direccionamiento Indexado Absoluto

En este tipo de direccionamiento nos encontramos con dos casos distintos, la diferencia entre ellos radica en que exista salto de página o no. Sino se sobrepasa el límite de la página, el resultado no origina acarreo. Este modo de direccionamiento resulta el más generalizado entre los indexados y se puede realizar por medio de los índices X e Y.

Direccionamiento por acumulador

Este tipo de direccionamiento sólo se da cuando la instrucción sólo implica la participación del Acumulador.

Operando		Código Op
A	ROR	6A

la posición de memoria que contiene ADL. Una vez que se recoge ADL, el contador del programa se incrementa hasta la siguiente posición de memoria que contiene ADH. Los modos de direccionamiento son absoluto y absoluto indirecto.

Ejemplo: JMP (Modo de direccionamiento absoluto).

Direcciones de memoria	Datos	Comentarios
0310	JMP	Salto a la posición 6000
0320	00	Nuevo byte del PCL
0330	60	Nuevo byte del PCH
6000	COD. OP.	Instrucción siguiente

JSR: Salto a una nueva posición del programa, guardando la dirección de retorno. Operación PC + 2 ↓, (PC + 1) → PCL. (PC + 2) → PCH.

También se suele indicar de esta forma:

(PC—1) → PCL
(PC—2) → PCH
(PC—3) → STACK

Salta a la dirección de una subrutina, dejando constancia de donde se rompió la secuencia para luego regresar a PC—3. Decrementa el valor del puntero de pila *(stack)*. El modo de direccionamiento es absoluto.

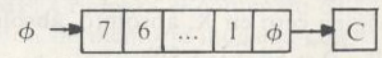
LDA: Esta instrucción carga el acumulador (A) con el contenido de la memoria especificado en la dirección del operando. Operación: (M)→ A. Cuando se efectua la instrucción LDA, los datos son transferidos de la memoria al acumulador y almacenados en él. Se afectan los indicadores N y Z. Es una instrucción del grupo cero y por tanto, tiene todos los modos de direccionamiento de este grupo.

LDX: Carga el registro de índice X con el contenido de una posición de memoria. Operación: (M) → X.

Afecta a los indicadores N y Z. Los modos de direccionamiento son: inmediato, absoluto, página cero, absoluto indexado por Y, página cero, indexado por Y.

LDY: Carga el registro de índice "Y" con el contenido de una posición de memoria. (M) → Y. Igual a la anterior.

LSR: Desplazamiento lógico a la derecha. Mediante esta instrucción se desplaza un bit a la derecha, sobre la información que contenga el acumulador o una posición determinada de memoria. La operación es:



El indicador "N" siempre está a cero y el indicador "Z" se pone a "1" si el resultado del desplazamiento es cero. Los modos de direccionamiento son: acumulador, página cero, X, absoluto, y absoluto X.

NOP: Esta instrucción significa no operar. El único resultado es el avance del contador de programa en una unidad. Tiene una duración de dos ciclos de máquina. Una de las posibles posibilidades de esta instrucción es la de poder programar intervalos de espera en el programa. El modo de direccionamiento es implícito.

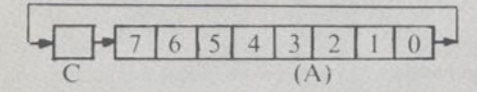
ORA: Esta instrucción realiza la operación lógica "OR" entre el contenido del acumulador y la posición de memoria especificada (A) V (M) → A o también (A) + (M) → A.

La instrucción se realiza bit a bit y el resultado se almacena de nuevo en el acumulador. Pertenece al grupo cero y por tanto los modos de direccionamiento son los de este grupo.

Ejemplo:

LDA 1110 x111 x ex "O" ó "1"
ORA 0000 1000
STA 1110 1111

MODOS DE DIRECCIONAMIENTO



Direccionamiento Implícito

Con este tipo de direccionamiento las instrucciones tienen tan solo un *byte*, en el que está implícita la función o funciones a realizar, así como el operando.

Las instrucciones con direccionamiento implícito hacen referencia a funciones internas en el microprocesador, como poner cero o a unos ciertos registros internos, transferir datos entre ellos, incrementarlos, etc.

Ejemplo:

	Operando		Código OP
C ← 0	Acarreo	CLC	18

Con esta instrucción, ponemos a cero el indicador "C" de acarreo.

Direccionamiento Relativo

Este modo de direccionamiento se usa para las instrucciones de bifurcación.

Las instrucciones con este tipo de direccionamiento, tienen dos *bytes*. El segundo es un número que se añade al contador de programa para provocar una bifurcación. Esta bifurcación se llama condicional si para que se ejecute hace falta que se cumpla alguna condición, condición que casi siempre es referente al estado de los bits del registro de estado. Las instrucciones que utilizan el direccionamiento relativo son: BCC, BEQ, BHI, BNE, BPL, BCS, BVC y BVS.

Familia 6500

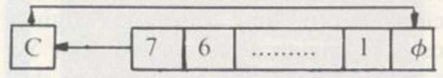
PHA: Con esta instrucción se transfiere el acumulador a la pila *(stack)*. Operación: A ↓. La flecha (↓) significa transferir a la pila. No se afectan ninguno de los indicadores y el modo de direccionamiento es implícito. PHA es una instrucción de un solo *byte*.

PHP: Mediante esta instrucción se transfiere el registro de estado (P) al *stack*. Se transfiere el contenido del registro de estado, a la pila sin alterarlo. Operación: P ↓. PHP es una instrucción de un solo *byte* y su modo de direccionamiento es implícito. No se afecta ningún indicador de *flags*.

PLA: Sacar el acumulador de la pila *(Stack)*. Operación: A ↑. Esta instrucción suma 1 al valor actual del puntero de pila y se emplea para cargar el siguiente contenido de ella en el acumulador. Sólo afecta a los indicadores de N y Z, siendo esta instrucción de un solo *byte*. El modo de direccionamiento es implícito.

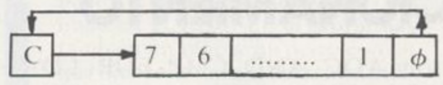
PLP: Mediante esta instrucción transferimos el contenido de la pila al registro de estados "P". Operación: P ↑. Si mediante la instrucción PHP transferimos el registro de estado "P" a la pila con PLP realizamos la operación inversa. No afecta indicadores y su modo de direccionamiento es implícito.

ROL: Rotación de un bit a la izquierda. Operación:



Esta instrucción, rota un bit del contenido del acumulador o de una posición de memoria, con bit de acarreo "C". Debido a su ejecución, modifica a los indicadores N, Z, y C. Tiene los modos de direccionamiento siguientes: acumulador, página cero, página cero en X, absoluto, absoluto en X.

ROR: Rotación de un bit a la derecha. Operación:



En esencia es lo mismo que la anterior, pero a la derecha.

RTI: Vuelva o retorno desde la interrupción. Operación: P↑, PC↑. Mediante esta instrucción volvemos al programa en curso después de haberse producido una "interrupción". (En los siguientes capítulos hablaremos detenidamente de las interrupciones).

El modo de direccionamiento es implícito.

RTS: Retorno desde la subrutina. Operación: PC ↑, PC + 1 → PC. Esta instrucción carga los contenidos alto y bajo del contador de programa desde la pila e incrementa el contador de programa para que apuntes a la instrucción siguiente. Es decir, salta a la posición que haya en la JSR. No afecta indicadores y su direccionamiento es implícito.

SBC: Resta del contenido del acumulador, el contenido de la memoria (operando) y el inverso del bit de acarreo, almacenando el resultado de nuevo en el acumulador. Operación: (A) - (M) - C → A.

Se afectan los indicadores N, Z, C, V. Pertenece al grupo uno en cuanto a los modos de direccionamiento.

SEC: Activar el indicador de acarreo. Operación: 1 → C. Pone el indicador "C" (Carry) a "1". Normalmente esta instrucción precede a la instrucción SBC. No afecta indicadores y su direccionamiento es implícito.

SED: Activar el modo decimal. Operación: 1 → D.

Pone el indicador "D" (Decimal) a "1". Todas las operaciones aritméticas que se realizan a continuación de esta instrucción, realizan su función en modo binario. No afecta indicadores y su direccionamiento es implícito.

SEI: Activa la prohibición de interrupción. Operación: 1 → I. Pone el

Ejemplo:	Operando	Código OP.	
(PC) ← (PC) + 5	5	BEQ 5	F0

Direccionamiento Indirecto

Este modo de direccionamiento sirve para resolver algunos problemas, donde la dirección efectiva se obtendrá por cálculos durante la ejecución de un programa, y por tanto, el programador no sabe de antemano el valor que tomará esa dirección. En este modo, se proporciona una dirección cuyo contenido y el de la siguiente, cargan al contador de programa dando lugar a un alto en el programa.

Ejemplo:	Operando	Código OP.	
(PC) ← (2830) y (2831)	2830	JMP 2830	6C

Direccionamiento Indexado Indirecto

Este modo de direccionamiento, resulta muy útil para la obtención de datos de una lista de direcciones. Se basa en disponer de una tabla de direcciones en la página cero, que podemos explorar secuencialmente usando un índice. Las instrucciones que disponen de este modo son: ADC, AND, CMP, EOR, LDA, ORA, SBC, y STA.

Direccionamiento Indirecto Indexado

Este modo tiene utilidad cuando necesitamos buscar un dato entre varios. El *byte* que sigue al código de operación apunta a una posición de la página cero cuyo contenido se suma con "Y", para obtener el *byte* de menor peso de la dirección del operando. Si hay acarreo en la suma, este

Familia 6500

indicador "I" (Interrupción) a "1", no permitiendo actuar a la línea IRQ. No afecta indicadores y su modo de direccionamiento es implícito.

STA: Almacenar el acumulador en una posición de memoria. Operación: (A) → M. Esta instrucción almacena enla dirección de memoria especificada en el operando, el contenido del acumulador. Pertenece al grupo uno y no afecta indicadores.

STX: Almacenar el contenido del registro de índice X, en una posición de memoria. Operación: (X) → M. No afecta indicadores. Transfiere el valor del registro X a la posición de memoria direccionada. Los modos de direccionamiento son: absoluto, página cero Y, página cero indexada por Y.

STY: Almacenar el contenido del registro de índice Y, en una posición de memoria. Operación: (Y) → M. Es igual al anterior.

TAX: Transferir el contenido del acumulador al registro X. Operación: (A) → X. Afecta a los indicadores N y Z. Su modo de direccionamiento es implícito.

TAY: Transferir el acumulador al registro Y. Operación: (A) → Y. Afecta a los indicadores N y Z y el modo de direccionamiento es implícito.

TYA: Transferir el índice Y al acumulador. Operación (Y) → A. Afecta a los indicadores N y Z. El modo de direccionamiento es implícito.

TSX: Transferir el puntero de pila al índice X. Operación: S → X. Afecta a los indicadores N y Z. Modo de direccionamiento implícito.

TXA: Transferir el registro de índice X al acumulador. Operación: (X) → A. Afecta a N y Z y su direccionamiento es implícito.

TXS: Transferir el índice X al puntero de pila. Operación: X → S. Afecta solamente al contador de la pila. El direccionamiento es implícito.

MODOS DE DIRECCIONAMIENTO

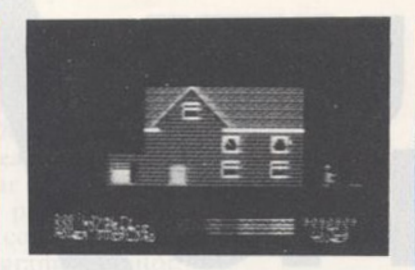
se suma a la siguiente posición de la página cero para obtener el *byte* de mayor peso. Las instrucciones que usan este modo de direccionamiento son: ADC, AND, CMP, EOR, LDA, ORA, SBC, y STA.

M. A. F.

PROGRAMA: HOVER BOWER
TIPO: JUEGO
DISTRIBUIDOR: INDESCOMP
FORMATO: CINTA DE CASSETTE
COMPUTADOR: COMMODORE 64 CON JOYSTICK

Este juego se encuadra dentro de la línea de los juegos de acción, aunque saliendo de la línea de los marcianitos típicos, a la que estamos acostumbrados. En este caso tenemos un jardinero que debe segar un jardín con un cortacésped en el menor tiempo posible.

La cinta con el programa se carga sin dificultad. A continuación aparece un menú desde el cual se empieza el juego. Debemos indicar que este juego (al igual que muchos otros existentes en el mercado) necesita que un *joystick* conectado al *port* 1, aunque este periférico sea bastante común hay personas que no lo tienen, por lo que no estaría de más tener la



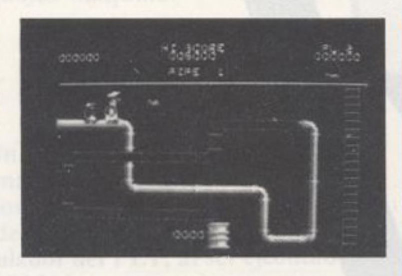
posibilidad de que se pudiese jugar con el teclado.

Al empezar el juego aparece el jardinero, que se dirige a una casa a coger una segadora. Después aparece un plano del jardín, compuesto por el césped, que debemos segar. Además hay flores (que debemos respetar) y setos que nos impiden el paso. En el jardín hay también un niño y un perro con los que debemos evitar encontrarnos. Si chocamos con el perro se nos trabará la máquina y perderemos tiempo y si el niño coge la segadora se escapa con ella, debiendo ir nosotros a por otra a la casa (hasta un máximo de tres). Cuando hayamos segado toda la hierba aparecerá un nuevo jardín en el que deberemos repetir el mismo proceso. Durante el desarrollo del juego van apareciendo divertidos mensajes en la parte inferior de la pantalla (aunque en inglés). Si por equivocación segamos las flores aparecerá una tercera persona que también nos querrá quitar el cortacésped, para evitar esta complicación añadida, debemos bordear las flores, a menos que sea extremadamente necesario. El botón de disparo tiene una característica especial, que consiste en que nuestro hombre se pondrá a gritar asustando al perro y, en menor medida, a los niños. Una vez que terminemos con el jardín el programa nos dará bonos por la lealtad del perro, ésta que es máxima al principio e irá disminuyendo cada vez que le gritemos.

PUNTUACION:
ADICCION: 6
PRESENTACION: 5
GRAFICOS: 8
ACCION: 6

PROGRAMA: PIPELINE
TIPO: JUEGO
DISTRIBUIDOR: ABC SOFT
FORMATO: DISCO Y CASSETTE
COMPUTADOR: COMMODORE 64

Este es uno de los primeros juegos que aparecen en *diskette* para el 64, ya que las casas de *software* no le habían prestado mucha atención, por ser un periférico caro y por tanto no muy común. En el programa somos los capataces de un oleoducto que termina en un barril. Después de dar innumerables vueltas, nuestra misión consiste en que el petróleo llegue al barril y lo llene. Para lograrlo debemos conducir a los obreros hasta los lugares donde esté cortado, para que lo arreglen. Estos cortes los producen los objetos lanzados desde la parte superior por unos hombres, asimismo hay arañas, que se lanzan al oleoducto y si nos cogen harán que caigamos, perdiendo una vida. Si cogen al obrero se caerá, con lo que tendremos que volver al principio



de la tubería a buscar otro que continúe el trabajo.

El juego se carga sin dificultad y posee una presentación muy bien hecha, haciéndose notar unos gráficos trabajados. Se puede jugar usando el teclado o un *joystick*, y posee la agradable particularidad de que si usamos el teclado podemos elegir entre dos grupos distintos de teclas para poder manejarlo. Una vez puesto en marcha comprobamos que los gráficos siguen teniendo la misma calidad excepcional que durante la presentación. Nuestro capataz aparece en la parte superior izquierda, al principio del oleoducto, junto con el obrero. Este le sigue a todos lados excepto cuando encuentra un lugar donde el tubo está cortado. En ese caso, se detendrá para arreglarlo, mientras nosotros podemos ir a otro lado y volver a buscarlo cuando acabe la reparación. En la parte derecha hay una escalera por la que suben los hombres y las arañas. Los hombres al llegar arriba tiran clavos y que cortan la tubería y las arañas caen a la tubería y empezando a recorrerla. Nuestro hombre dispone de una pistola con la que puede matarlos y también a los hombres cuando suben por la escalera o (en el caso de las arañas) cuando están en la tubería. Una vez que llenamos el barril pasaremos a tuberías sucesivas en las cuales se complicarán las cosas. Por ejemplo aparecerá una gran araña inmune a las balas y que tenemos que evitar a toda costa, y también protecciones en la escalera de modo que sea más difícil alcanzar a los animales cuando suben.

PUNTUACION:
ADICCION: 6
PRESENTACION: 8
GRAFICOS: 8
ACCION: 6

UTL 6440
Utilidades para el programador



Programmer's utilities UTL - 6440 (utilidades del programador) es un disco que contiene 12 programas de utilidad creados por **Commodore Business Machines** para ayudar al usuario del **64,** proporcionándole una serie de programas que tratan diversos aspectos del ordenador.

Los programas disponibles en el disco son los siguientes:

1. **Change disk**
2. **Copy-all46**
3. **Hex dump**
4. **Load addr**
5. **Supermon64.VI**
6. **Char editor**
7. **Sprite editor**
8. **Dos wedge**
9. **Pet emulator**
10. **1541 backup**
11. **Editor 64**
12. **Sidmon**

En el manual, los programas van agrupados en 4 secciones, según el tipo de aplicación a que va destinado cada uno. Las secciones son: Utilidades, gráficos, sonido y ayudas a la programación en BASIC. Vamos a ver qué posibilidades ofrece cada programa.

UTILIDADES

Esta sección comprende 8 programas de utilidades, que hacen referencia a diversos aspectos del ordenador como por ejemplo las unidades de disco, el lenguaje máquina o la emulación del ordenador **PET.**

CHANGE DISK

Es el primer programa (la traducción es "cambia el disco") que permite al usuario cambiar el número de dispositivo de cualquier unidad de *diskette*. Normalmente este número es 8. La idea en que se basa este cambio en el número del dispositivo es poder emplear 2 unidades de disco, por ejemplo para copiar ficheros con el programa **Copy-All 64.**

COPY-ALL64

Copia-todo64 es un programa para copiar uno o más ficheros (o programas) de un *diskette* a otro. La copia se puede efectuar entre dos unidades de disco diferentes o empleando una sola unidad. Lo que ocurre en este segundo caso es que hay que cambiar sucesivamente de disco "fuente" a disco "destino" en la unidad, según se va efectuando la copia. El programa permite copiar todos los ficheros de un disco, o solo ciertos ficheros, que se pueden seleccionar automáticamente especificando ciertos criterios, como por ejemplo que empiecen con la letra "A". El programa se encarga asimismo de formatear los discos virgen.

1541 DISK BACKUP

Se trata de otra utilidad que permite copiar discos. En este caso sólo está previsto el empleo de una unidad de disco. La diferencia entre este programa y el anterior está en que el **1541 Disk backup** copia los discos por pistas y sectores, no buscando programa a programa. En realidad son posibles dos modalidades de copia. La primera sólo copia las áreas escritas del disco, según viene especifican-

do en el BAM del propio *diskette*. Con la segunda modalidad se copia el *diskette* entero, pista a pista y sector por sector, que es por lo general más lenta que la anterior. En teoría este programa de copia permitirá copiar cualquier *diskette*, esté protegido o no, aunque en realidad siempre hay formas de proteger un disco para que no pueda ser copiado por un determinado programa copiador, lo mismo que siempre hay un programa copiador que puede saltarse una protección determinada.

DUMP

El programa **DUMP** (Volcar) vuelca en pantalla un fichero cualquiera del contenido en el *diskette*, visualizándolo en hexadecimal, mostrando 10 bytes por línea. De esta forma se puede "andar" por entre los ficheros en modo hexadecimal, lo que puede ser útil e interesante, especialmente para los amantes del lenguaje máquina.

LOAD ADDRESS

Load Address (carga dirección) permite conocer la "dirección de carga" de cualquier fichero (siempre que sea un programa), es decir que permite conocer la dirección de memoria donde comenzaba el programa cuando fue guardado en el disco. Esto puede ser útil por ejemplo para manejar programas en lenguaje máquina que no sean reubicables, esto es que sólo funcionan bien cuando están cargados en determinadas posiciones de memoria. El programa muestra esta dirección de carga en decimal.

SUPERMON64.VI

Otro de los programas de este disco de utilidades es **Supermon64.VI.** Consiste en un monitor de lenguaje máquina, es decir un programa que permite trabajar directamente en lenguaje máquina (en realidad en hexadecimal), haciendo posible el acceso directo a posiciones de memoria, a los registros de microprocesador, desensamblaje de instrucciones, etc. Su misión es servir de ayuda en el desarrollo y depuración de programas en lenguaje máquina.

Después de cargar el programa y teclear RUN, aparece un punto en lugar que ocupaba el cursor rectangular, pudiendo accederse al conjunto de comandos que constituyen este programa monitor.

El conjunto de comandos permite entre otras cosas:

— Cargar programas desde *cassette* y *diskette*.

— Ver y modificar el contenido de posiciones y memoria, así como el de los registros del procesador.

— Buscar cadenas de caracteres ASCII.

— Transferir el contenido de bloques de memoria de un lugar a otro.

— Ejecutar programas en lenguaje máquina.

— Introducir instrucciones en ensamblador y desensamblar instrucciones.

En definitiva, un conjunto no muy amplio de comandos pero suficiente para desarrollar pequeñas rutinas en lenguajes máquina.

PET EMULATOR

Un programa emulador es un programa que hace que un cierto ordenador se comporte como si fuera otro modelo distinto. En nuestro caso, el **Emulador del PET,** al ser ejecutado en el **CBM64,** hace que éste se comporte como si fuera un **PET.** Esto se consigue mediante una reconfiguración de la memoria y una traducción de instrucciones. Por ejemplo, la memoria de pantalla está situada en distintas direcciones en ambos ordenadores:

Dirección de la memoria de pantalla:

PET
$8000-8400
CBM64
$0400-0800

Al utilizar el emulador, el usuario ve la memoria de pantalla en las direcciones correspondientes al **PET** y al ejecutar un programa para **PET,** los PEEK y PUKE enviados a la memoria de pantalla funcionan correctamente.

DOS WEDGE

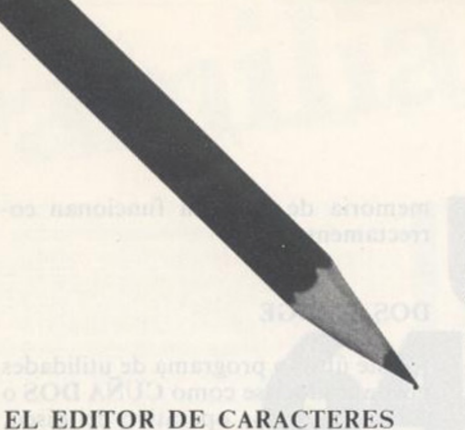
Este último programa de utilidades puede traducirse como **CUÑA DOS** o cuña del sistema operativo del disco. El nombre de "cuña" proviene del hecho de que este programa actúa a modo de cuña, insertándose en el sistema operativo del **CBM64.** De este modo al pulsar cualquier tecla o escribir cualquier comando, el **DOS WEDGE** comprueba si la orden es para él y si no, se la deja al sistema operativo, con lo que todo funciona normalmente. Con esto se consigue añadir nuevos comandos al BASIC del **64,** comandos que se refieren al disco. En realidad no se añaden comandos, sino que se simplifican, es decir, este programa no permite hacer con el disco nada que no se hiciera previamente, pero consigue que cualquier comando del disco pueda ejecutarse pulsando una sola tecla, lo que supone un gran ahorro de tiempo en las operaciones con *diskettes*.

GRAFICOS

En un conjunto de programas de utilidades no podían faltar dos programas característicos, para el manejo de gráficos y caracteres, como son un **EDITOR DE CARACTERES** y un **EDITOR DE SPRITES.** Estos dos programas constituyen la sección dedicada a los gráficos.

Editores de caracteres y de Sprites hay muchos, incluso es corriente encontrar programas de este tipo publicados en las revistas. Lo que distingue a unos de otros es el conjunto de comandos que ofrecen para el diseño y almacenamiento de caracteres y Sprites, y la mayor o menor facilidad en el empleo de estos comandos. Todo ello define la potencia de estos programas y nos permite decidir si justifican el dinero que pagamos por ellos.

Vamos a ver cuáles son las posibilidades de cada uno de estos dos programas.



memoria donde funcionan correctamente.

EL EDITOR DE CARACTERES

Antes de revisar el editor de caracteres, recordemos algunas de las características del ordenador. En el 64 es posible trabajar con 7 juegos de caracteres diferentes. Dos de estos juegos van incorporados en ROM (memoria no volátil) y son los que se utilizan normalmente en los modos gráfico y minúsculas. Además de estos dos, se pueden programar otros cinco juegos de caracteres pero que, por esta situados en posiciones correspondientes a la memoria RAM (memoria volátil), hay que cargar cada vez que se enciende el aparato. Los juegos de caracteres, cada uno de los cuales ocupa 2048 (2K)bytes de memoria, son los siguientes:

JUEGO N.°	SE ACTIVA CON
1	POKE 53272, 19
2	POKE 53272, 21
3	POKE 53272, 23
4	POKE 53272, 25
5	POKE 53272, 27
6	POKE 53272, 29
7	POKE 53272, 31

DIRECCION EN MEMORIA	TIPO DE MEMORIA
2048 - 4095	RAM
Modo gráficos	ROM
Modo minúsculas	ROM
8192 - 10239	RAM
10240 - 12287	RAM
12288 - 14335	RAM
14336 - 16383	RAM

Veamos ahora que nos ofrece el editor de caracteres.

El programa, una vez cargado desde la unidad de *diskette*, dispone de dos modos de trabajo, que se denominan de SELECCION DE CARACTERES y de EDICION DE CARACTERES.

El primer modo nos permite seleccionar y visualizar cualquiera de los 7 juegos y elegir cualquier caracter de cualquier juego para modificarlo. Además permite cargar desde el disco, o almacenar en él, cualquier juego de caracteres, lo que permite tener almacenados un gran número de juegos diferentes para utilizarlos cuando convenga. (El disco incluye un juego de caracteres denominado COMPUTER.SET 5).

Por último, este modo incluye comandos para cambiar los colores del fondo y reborde de la pantalla.

COMANDOS DEL MODO SELECCION

COMANDO	QUE HACE
1	Selecciona y muestra en pantalla el juego de caracteres 1.
......	
7	Selecciona y muestra en pantalla el juego de caracteres 7.
CTRL-N	Muestra 64 caracteres del juego en curso.
CTRL-B	Proporciona 16 colores para el fondo
CTRL-E	Proporciona 16 colores para el borde.
CTRL-L	Carga un juego desde el disco.
S	Almacena un juego en el disco.

El modo de EDICION DE CARACTERES permite diseñar o modificar cualquier caracter e incorporarlo a cualquiera de los juegos de caracteres. Los comandos de este modo se utilizan para diseñar el caracter punto a punto, sobre una rejilla, y para emplear cualquiera de ellos basta pulsar una tecla. Los comandos sirven para:

— Encender o apagar cualquier punto de la rejilla, así como borrar o invertir todos los puntos de la misma.

— Moverse por la rejilla en cualquier dirección.

— Mover el caracter de la rejilla una posición en cualquiera de las cuatro direcciones arriba, abajo, derecha e izquierda o girarla 90 grados.

Cambiar el color del fondo o reborde.

— Asignar el caracter a un juego de caracteres.

EL EDITOR DE SPRITES

En el **Commodore 64,** los Sprites dan una gran flexibilidad a la hora de crear gráficos con movimiento, haciendo sencilla la animación de imágenes para juegos. Cada Sprite utiliza 64 bytes de memoria, lo que se denomina una página. Estos 64 bytes corresponden a una rejilla de 21 líneas de 3 bytes (24 bits) por línea, y es posible tener hasta 160 Sprites en memoria simultáneamente. Las posiciones de memoria asociadas a estos Sprites son las siguientes:

PAGINAS (1 SPRITE POR PAGINA)	DIRECCION DE MEMORIA	NUMERO DE SPRITES
32 - 63	2048 - 4095	32
128 - 255	8192 - 16383	128

La única pega estriba en que crear un Sprite es una labor tediosa, ya que después de dibujarlo sobre una rejilla hay que codificarlo para introducirlo en la memoria del ordenador. Como cada Sprite consiste en 64 bytes, cuando se utilizan varios, el número de bytes a codificar se hace muy grande, y resulta muy útil contar con un programa como este **EDITOR DE SPRITES.**

El **EDITOR DE SPRITES** es un programa que permite crear, modificar y almacenar Sprites de una forma sencilla. En realidad se encarga de toda la tarea de codificar el dibujo y almacenarlo en memoria o en *diskette,* con lo que el usuario sólo tiene que dibujar el Sprite, directamente sobre la pantalla.

Al ejecutar el programa aparece en

la pantalla una rejilla de 21 líneas × 24 columnas que albergará el Sprite, pudiéndose modificar uno existente. El diseño se lleva a cabo mediante una serie de comandos de una sola tecla (o CTRL + una tecla) y que permiten hacer lo siguiente:

— Encender o apagar cualquier punto de la rejilla, además de borrar o invertir todos los puntos a la vez.

— Desplazar el cursor a derecha, izquierda, arriba y abajo con las teclas de cursor, a la línea siguiente con RETURN o al extremo superior izquierdo con HOME.

— Mover el Sprite en la rejilla una posición hacia arriba, abajo, derecha o izquierda o bien girarlo 90 grados.

— Seleccionar una página (es decir un Sprite) entre los 160 posibles, y acceder al Sprite siguiente (con la tecla +) o al anterior (con la tecla —).

— Cambiar los colores del Sprite, fondo y reborde.

— Modificar el tamaño por expansión o contracción horizontal, o vertical, de forma independiente.

— Cambiar de modo de alta resolución a modo multicolor y viceversa.

— Mostrar varios Sprites a la vez, hasta 8 por pantalla, además con la posibilidad de ver el Sprite quieto o en movimiento. En este último caso con la posibilidad de variar la velocidad a la que se mueve.

— Por último se puede almacenar un conjunto de Sprites en *diskette* o ser cargados desde la unidad de *diskette* a la memoria del ordenador, cada uno con un nombre, para identificarlo.

SONIDO

Como se sabe, el BASIC utilizado en el **Commodore 64** no incluye instrucciones específicas para el manejo de sonidos. Hay que utilizar una serie de POKE y modificar todo un conjunto de registros, para que aquello suene. La verdad, es una lástima, porque el chip 6581 (Sound Interface Devide o SID) es un verdadero sintetizador de música integrado, con grandes capacidades para generar todo tipo de sonidos, pero al no ser fácil de manejar, queda en general bastante desaprovechado. Existen programas, y el **SIDMON** es uno de ellos, que actúan de intermediarios entre el *chip* SID y el programador, para simplificar los pasos de la generación de sonidos, de forma que en lugar de tener que utilizar diversos POKE para modificar aspectos de los sonidos, basta con pulsar una tecla. Después de cargar el **SIDMON** desde el *diskette*, aparecen en la pantalla todos los comandos que pueden utilizarse. Al pulsar la tecla correspondiente a cada comando, que es la letra

del nombre del comando que aparece en inverso, se modifica alguno de los parámetros del sonido, por ejemplo la frecuencia o la forma de onda. Modificando cada uno de los parámetros se pueden explorar uno a uno todos los sonidos que es posible generar. Si interesa recordar alguno de ellos, basta con anotar en un papel el valor de los registros del *chip* SID, correspondientes a dicho sonido ya que este valor también aparece en la pantalla junto al comando asociado.

Los comandos del **SIDMON,** junto con el efecto que producen, son los siguientes:

F1, F3, F5	Seleccionan el canal (o voz) con que se trabaja, de las 3 voces existentes.
+, —	Se usan para aumentar o disminuir el valor contenido en un registro y hay que usarlos después de la tecla correspondiente a dicho registro. También actúan como interruptores ON/OFF.
M, D	Para multiplicar o dividir por 2 la frecuencia o el ancho de los impulsos rectangulares.
F	Sirve para modificar la frecuencia de los sonidos.
P	Modifica la anchura de los impulsos rectangulares.
A, D, S, R	Determinan los tiempos de ataque, caída, sostenimiento y relajación.
G	G seguido de + comienza la fase de ataque, caída y sostenimiento. Al pulsar — comienza la relajación.
Y, I	Sincronismo y Ring. Sincronizan en frecuencia dos canales o se produce una modulación entre dos canales.
1, 2, 3, 4	Determinan las formas de onda.
T, U, E	Determinan si el filtro actúa o no, su frecuencia de corte y si se produce o no énfasis.
V	Fija el volumen.

AYUDAS A LA PROGRAMACION BASIC

EDITOR 64

Editor 64 es un programa editor de datos en pantalla, es decir que proporciona una serie de comandos para formatear en la pantalla, la introducción de datos. Esta suele ser una labor complicada a la hora de escribir un programa en BASIC y normalmente lleva bastante tiempo y requiere muchas instrucciones.

Editor 64 proporciona una serie de comandos de una sola tecla, que permiten introducir los datos por la pantalla en campos definidos previamente. Los datos, una vez introducidos, se guardan en una matriz SC$, y pueden ser utilizados posteriormente por un programa. Los mismos comandos para el formateo de campos e introducción de datos también pueden utilizarse desde dentro de un programa, por lo que en realidad es como si se hubieran añadido nuevas sentencias al BASIC.

Hay 3 tipos de campos para la introducción de datos que son:

1. Campos alfanuméricos, entrada de izquierda a derecha.
2. Campos alfanuméricos, entrada de izquierda a derecha y editables.
3. Campos numéricos, entrada de derecha a izquierda.

En cuanto a los comandos que se añaden al BASIC, podemos destacar los siguientes:

&c para cambiar colores.

&l para dibujar líneas horizontales.

&e para entrar en modo edición y editar cualquiera de los campos.

&t y &b para visualizar 2 líneas de *status* en pantalla con informaciones diversas como el modo de trabajo.

El **Editor 64** constituye pues un conjunto de herramientas destinadas a facilitar la escritura de programas en BASIC, específicas para el formateo en la introducción de datos por pantalla.

EL MANUAL

Es sin duda alguna el aspecto más flojo de este paquete. Para empezar digamos viene en inglés y desconocemos si habrá o no traducción al castellano, esto pudiera ser un grave inconveniente para los principiantes. Pero además, este manual, sigue fiel a la línea de **Commodore** con respecto a sus manuales, incluye muy poca información. Los ejemplos brillan por su ausencia, hay muy pocas aclaraciones y detalles sobre el funcionamiento de los diversos programas también se hecha de menos la existencia de algunas hojas con los comandos resumidos para consultas rápidas, para recordar cualquier comando hay que ir buscándolo a través del manual. Tampoco existe índice de temas ni de comandos. En definitiva, un manual bastante regular y que desmerece de las interesantes posibilidades que ofrecen los paquetes de herramientas para obtener una programación más sencilla y más eficaz.

Roberto Menéndez



La nueva revista para usuarios del ZX-81 y SPECTRUM

Programas/Juegos/Montajes/Código Máquina

AÑO 1 · No. 6 / MAYO · 84 · 200 Ptas.

ZX

REVISTA PARA LOS USUARIOS DE ORDENADORES SINCLAIR

Construya su propio juego

¡Ya está a la venta!
Cómprela en su quiosco o solicítela a:

Bravo Murillo, 377
Tel. 733 74 13
MADRID-20

Tamaño real: 19,5 x 26,5

LASER 2001
200



Unidad principal: CPU. Z80A, a 3, 58MHZ, 4 K RAM, 16 K ROM, 8 colores en pantalla. Sonido. Gráficos alta resolución: 128 x 64. Periféricos: Ampliaciones a 16 K y 64 K. Interface Impresora Centronics. Impresora 4 colores. Joysticks.



Unidad principal: CPU. 6502A a 2MHZ. 32 KRAM. 16KROM. 16 colores en pantalla. 4 canales de sonido. Gráficas alta resolución: 256X192. Periféricos; Ampliaciones de memoria. Interface impresora Centronics. Joysticks. Floppy disk. Adaptador juegos de Colecovisión y Atari.

VTECH

IMPORTADOR EXCLUSIVO
intercomsa
AVDA. BRASIL, 7 - MADRID-20
TELEF. 455 60 43 TELEX: 43980 ICOE-E

DISTRIBUIDOR EXCLUSIVO CATALUÑA
H. E. C. I. S. A.
AVD. INFANTA CARLOTA, 80-82, int. 4 - BARCELONA-29
TELEF. 230 62 47 TELEX: 51506 ZAZU